Cloud based emails boundaries and vulnerabilities

Chris Masone and Sean Smith
Department of Computer Science
Dartmouth College
cmasone,sws@dartmouth.edu

***Abstract*— When users' mental models don't match the way the underlying systems work, problems can arise. For human-based security systems to be effective, we believe that is important to identify the tasks involved at which humans excel (and at which computers do not), and then design the system accordingly. To demonstrate this principle, we are building *Attribute-Based, Usefully Secure Email (ABUSE)*, a system that leverages users by enabling them to build a decentralized, non-hierarchical PKI to express their trust relationships with each other, and then to use this PKI to manage their trust in people with whom they correspond via secure email. Our design puts humans into the system—to do things that humans are good at but machines are not—at both the creation of credentials as well as the interpretation of credentials. In this paper, we discuss why secure email is an interesting proving ground for our design ideas, set out the architecture of the system, and relate our early experiences in testing our user interface on real humans.**

I. Introduction

In the current era of ubiquitous network connections, a wide array of software has come into existence to help users secure various aspects of their computer use. As users work with these pieces of software, they must constantly make decisions regarding the integrity and authenticity of incoming information and requests for personal data. This software often attempts to help users protect themselves through a variety of user interface elements — icons, dialog boxes, text elements and so forth. Given the current state of affairs with respect to computer security, it is clear that something has gone awry.

Our own experience, as well as work done by other researchers [1]–[3], supports the idea that problems often arise when software behaves in a way contrary to the mental model suggested by its user interface. Moreover, when software attempts to help humans make decisions, but is not capable of appropriately modeling how humans make those decisions, it may provide information that is insufficient, irrelevant, or even totally misleading. To bring human mental models and the behavior of systems into closer alignment, thus avoiding these problems, we believe software must be designed to explicitly leverage the people who use it; lay out the goals of the system, identify the tasks involved at which humans excel (and at which computers do not), and then design the system accordingly.

This research has been supported in part by the NSF (CNS-0448499) and by Intel. This paper does not necessarily reflect the views of the sponsors.

To demonstrate this design principle, we have chosen to build *Attribute-Based, Usefully Secure Email (ABUSE)*, a system that leverages users by enabling them to build a decentralized, non-hierarchical PKI to express their trust relationships with each other, and then use this PKI to manage their trust in people with whom they correspond via secure email. Our design puts humans into the system—to do things that humans are good at but machines are not—at both the creation of credentials as well as the interpretation of credentials. By doing so, we hope this system can overcome the failings of approaches based on standard PKI.

Because "trust" is a heavily overloaded term across several fields, Section II will present and explicate how we mean to use this term. Section III will provide background on S/MIME email, the application we have chosen to use to explore human trust issues as well as some motivation for that choice. Section IV discusses the related field of *Trust Management (TM)* and why it is not an appropriate solution for the problems we consider. In Section V, we will discuss the ABUSE architecture and our experiences designing one of the system's user interfaces. Section VI will discuss our future work with the system, related work will follow in Section VII and we will wrap up with concluding remarks in Section VIII.

II. Humans and their trust

To borrow from both sociological literature and and technical literature, we will consider *trust* to be the willingness of an entity to undertake a potentially dangerous action on behalf of a second entity as a result of a set of shared expectations between the two [4], [5]. There are two components to this set of expectations:

- Background expectations, the assumptions defined by a "world known in common" [6], and
- Constitutive expectations, the parameters of the particular situation [7].

According to [4], there are three ways to create this context between two parties:

- Process: using reputation and prior experience.
- Characteristic: using innate attributes, e.g. family background, gender or ethnicity.
- Institutional: using formal social structures, like certifications or membership in a professional organization.

It may be interesting to apply this framework to the myriad of methods in which humans interact through the Internet. However, we believe it will be more instructive and useful to apply this framework to some concrete examples from a single domain—email, since it's become the primary means of communication between humans in Internet settings.

III. S/MIME Email: A Partial Solution

To address email security and privacy concerns, many organizations in the commercial, federal and educational sectors have deployed S/MIME [8], [9], a secure email standard that leverages X.509 Identity Certificates [10] to provide message integrity and non-repudiation via digital signatures [11], [12]. In addition, these signatures often contain the sender's Identity Certificate, so all information contained therein is available to the recipient. For example, a digitally signed message sent by the first author would confirm that "Dartmouth College" believes his email address to be "Christopher.P.Masone@dartmouth.EDU", his name to be "Christopher P. Masone", and give a date after which Dartmouth no longer guarantees any of the above to be true. The signature would also contain his public key, which can then be used to validate the signature. *Pretty Good Privacy (PGP)* is another PKI-based email scheme which provides similar properties, but with more sporadic adoption.

In terms of our trust model, S/MIME can do one of two things for the recipient, depending on whether she has experience with the sender. If she knows the sender a priori, S/MIME can enable the recipient to leverage her trust in an institution to assure herself of the sender's identity and thus apply her process-based trust to the incoming message. If she has little or no prior experience with the sender, then S/MIME allows the recipient to extend some measure of institutionally-based trust to the sender. At least, that's the idea. However, both the literature and personal experience show that issues remain.

A. Familiar correspondents

S/MIME leverages X.509 ID certificates, which are minted by a Certification Authority (CA), often local to the user receiving the cert. In the above case, Dartmouth College's CA would have issued Chris his credential. If Chris sends signed email to someone outside of Dartmouth, most email clients will actually warn the recipient *not to trust that message*, unless she has configured her software to trust Dartmouth's CA [13]. The user is told *not* to trust someone that she *already does* because of a client configuration issue. To fix this, the recipient would have had to install the Dartmouth CA's certificate as a "trust root" in her email client. Standard clients come with a wide variety of trust roots pre-configured but, with many organizations deploying their own PKIs (especially educational institutions), this set is far from comprehensive.

Another interesting issue arises from the fact that standard S/MIME clients treat all installed trust roots as equal.[1] When an S/MIME signature is deemed valid, the client will display the same information to the user *regardless of which CA issued the credentials used to sign the message!* The first author leveraged this quirk, along with Thawte's Freemail CA and Dartmouth's name directory, to generate what appears at first glance to be legitimately signed S/MIME email from the College's president. The Freemail CA will allow a user to get a certificate for any email address over which he can demonstrate control. Dartmouth's name directory allows users to choose any nickname, even one close to the actual name of the President. Yes, the certificate used to sign this message was not from Dartmouth's CA, but this is only evident after some extra effort (our Director of Technical Services was reportedly taken in by our ruse).

In the case of familiar correspondents, S/MIME is supposed to help users leverage their trust in an institution to allow them to reliably extend their pre-existing process-based trust to each other. However, as we have shown, it is possible for an attacker to play several complex systems off each other and thwart this design.

[1]At least, all that are allowed to identify users for the purposes of signing email.

B. Unfamiliar correspondents

In large organizations, it becomes less likely that a sender and recipient knew each other prior to contact. Thus, an S/MIME signature verifying only the sender's name and email address would not be enough to help the recipient make a good decision. The signature is not expressive enough to allow humans to specify the right properties for conclusions in human trust settings. The authors, along with another colleague, previously explored S/MIME expressiveness problems in [14]. The following paragraphs summarize the classes from that report.

The first class of issues arises when users expect that a name, verified by a digital signature, equates to a person. In Dartmouth's Computer Science department, for instance, our grant manager shares a name with many other people in our Name Directory. They are distinguishable only by middle initial (not helpful) and department (since she is listed in a generic administrative unit, also not helpful). We don't care that mail is from "Joan B. Wilson"[2], we care that mail is from "the Joan Wilson who manages grants for Computer Science at Dartmouth."

A second class of issues arises when a name, verified by a digital signature, does not tell the user what they need to know. A senior colleague has preached the need for academic PKI to prevent a repeat of an incident at Yale in which someone forged mail from the Dean, canceling classes. Here at Dartmouth, we often receive "mail from the Dean" that is not from the Dean at all, but from one of the Dean's administrative assistants. Standard, hierarchical PKI doesn't help at all in this case, because a signature from "John Wilson" doesn't help unless the recipients know that "John Wilson" is the new assistant to the Dean, and is allowed to speak for her on such matters.

The final class of expressiveness issues shows up when the same property does not mean the same thing in different contexts. Take the case of a colleague who moved to another university, and was asked for an extension by a student who had an athletic event in which he needed to participate. Our colleague, used to Dartmouth where coaches are faculty or staff, gave permission, pending an email from the student's coach confirming the event. "John Wilson", who really was the coach, sent mail to confirm. "John Wilson" was also a student, happy to help his friend get out of work.

According to surveys we have conducted, problems such as these (and the other examples in [14]) are currently worked around by phone calls, searching the institution's website, checking a company directory, or just assuming that everything is fine. The "unmotivated user" property of security [1], which states that users will give up on behaving securely if it is too difficult or annoying, leads us to believe that the last case is the most likely. Therefore, relying upon people to check up on every suspicious email is not a winning strategy. Furthermore, our experiences in penetration testing and discussions with a security consulting firm [15] have shown that a lot of organizational information (especially about educational institutions) is accessible via the internet. This makes it more likely that an outside attacker would be able to craft messages that appear to be plausible, despite not being an insider. Given this, it becomes clear that a system which addresses the problems detailed above as a part of the normal workflow is desirable.

In terms of our trust model, in the case of unfamiliar correspondents hierarchical-PKI-based S/MIME is attempting to allow users to build institutional trust between each other. What's really going on here is that membership in a subculture is being established, eg. the "Member of the Dartmouth Community" subculture. This allows some kind of sphere to be defined in which the individual can be trusted. Standard S/MIME implementations can only establish membership in a fairly large subculture. The members of this group are not homogenous enough to clearly define an area in which all members should be trusted. If we can build a system that allows a smaller subculture to be defined (”Members of the PKI/Trust Lab”, or ”Sean Smith's PhD Students”), this makes it more likely that users will be able to come to useful trust conclusions.

Despite these issues, S/MIME *has* provided both message integrity and non-repudiation, as well as the sender's public key, provided that the recipient trusts the sender's CA and that the sender's private key has remained private. S/MIME, therefore, is a good starting point, and the public key in particular could provide a way to hook further contextual information about the sender into the message.

IV. Trust Management

At first glance, combining a Trust Management (TM) system with S/MIME seems to be an appropriate way to provide users with the extra contextual information that we wish to provide. Li et al. define *trust management* as "an approach to distributed access control and authorization, in which access control decisions are based on *policy statements* made by multiple principals" [16].

[2]All names have been changed to some variant of "John Wilson" for anonymity

Depending on the system, digitally-signed policy statements may be called *credentials* or *attributes*. These credentials must come from some party who is qualified to mint them. Also, when a request is made, credentials must be bound to that request. TM systems typically implicitly assume some type of public key infrastructure to provide these properties. Upon receipt by a TM engine, a request and its supporting credentials (possibly combined with other policy statements pulled from a local or remote credential repository) are then checked against the resource owner's trust policy (another set of policy statements). If the request satisfies the policy, authorization is granted.

Li and Mitchell defined a useful framework for discussing trust management systems, which "consists of three aspects: language, deduction and infrastructure." [16] A TM language, according to [16],

> has a mechanism for identifying principals, a syntax for specifying policy statements and queries, and a semantic relation that determines whether a query is true given a set of policy statements.

The deduction engine of a TM system implements these semantics, while the infrastructure provides support of the creation, maintenance and transport of policy statements. Several of the TM systems discussed here leverage the "logical programming" paradigm, while others choose not to base their approach on this model. In all cases, parties involved in the TM system can make assertions about the attributes possessed by other parties, as was touched on above. Resource owners, then, can express policies not only in terms of principals, but also in terms of these attributes. For instance, my policy could state that I trust the "Dartmouth" principal to grant a "student" credential to all currently enrolled students, and that anyone presenting such a credential issued by "Dartmouth" can access my "fun stuff to do in Hanover" files.

To implement a TM system atop email, we would need to

1) choose a policy language,
2) attach some credentials to digitally signed messages,
3) build some kind of policy-checking engine into an email client and, lastly
4) convince users to specify policies that accurately capture their trust behavior when reading email.

Before we can intelligently discuss such an implementation, we must first survey existing TM systems in greater detail.

A. Logic-Based Approaches

TM systems which leverage the "logical programming" paradigm either use some Prolog-like language to specify policy statements or design a new policy language that can be reduced to Prolog. We will illustrate this portion of the space by evaluating Delegation Logic (DL) [17], the Role-based Trust-management (RT) framework [16], [18], [19], SD3 [20], and Trust Policy Language (TPL) [21].

DL, SD3 and RT are all TM systems based on Datalog, a form of Prolog. They leverage the logical-programming paradigm to prove that a request, along with appropriate credentials, meets a given policy. SD3 and RT both provide some facility for retrieving credentials from non-local repositories, while DL does not. To use these systems, a user would have to specify his trust policy in one of these logical programming languages.

TPL is an XML-based TM language that can be reduced to Prolog. Like SD3 and RT, it also provides some facility for remote credential retrieval. Policy generation is once again done by hand. Though XML is a more accessible language than Datalog, a user would still need to be comfortable with programming to use TPL.

B. Other Systems

Other systems, such as PolicyMaker [22] and REFEREE [5], define their own policy languages which allow portions of policies to be defined by arbitrary programs written in any of several other fully-programmable languages. The KeyNote [23] work, a follow-on to PolicyMaker, specifies a simpler policy language that does not allow the policy writer to include arbitrary code.

REFEREE provides a very simple syntax for defining policy statements, based on s-expressions. However, much of the expressiveness of the system is provided by an *invoke* construct, which allows for the execution of arbitrary code. PolicyMaker also has a simple syntax for credentials and policies, and allows portions of policies to be defined in one of a set of scripting languages, which provide much of the expressiveness of the system. Thus, to do much that is complex, a user must again be a programmer.

KeyNote defines its own TM language. This language is reasonably expressive, though policies cannot be as expressive as in either system that allows for the inclusion of fully programmable languages. To use this system, a user would have to learn the KeyNote TM language, and then define his policy in those terms.

C. *Commonalities*

One common thread between all these systems is that they require the resource owner not only to be capable of writing computer programs, but also to be capable of boiling their trust decisions down to a consistently and mechanically applicable policy. This might be acceptable when managing trust between organizations (in a business-to-business type of relationship, for instance), where systems are managed by trained administrators and trust policies are often fully specified in contracts. However, if we are trying to build a TM system atop email to help regular end users decide how much to trust incoming messages from people they don't know, we cannot expect these non-programmers to learn a programming language just so that they can configure their email software. While it may be true that a trained administrator could construct some default policies to apply to all users, the real value of an email trust management system would be in capturing each individual's trust policy.

It is rare that humans implicitly trust every word that another human says. Usually, users choose to trust certain kinds of people to talk about certain kinds of things. Creating TM policies to govern this sort of thing would require the TM system to be able to comprehend incoming email. Doing Natural Language Processing (NLP) of this kind for without a corpus of material written by the sender is intractable [24]. Since we can't expect real users to program up their policies, can't prewrite them, and can't design a useful system without being able to comprehend arbitrary human language, a TM approach to helping end-users decide email trust does not seem appropriate.

D. *In Sum...*

Although S/MIME can verify that the message content hasn't changed, it only communicates institutional credentials at a very coarse granularity. TM is too complicated for recipients—and requires NLP that works. On the sender end, mechanical processes do seem sufficient to provide the right credentials for the context of that message and that sender. On the recipient end, mechanical processes do not seem sufficient to turn whatever credentials arrive into a reasonable trust decision.

V. Attribute-Based, Usefully Secure Email (ABUSE)

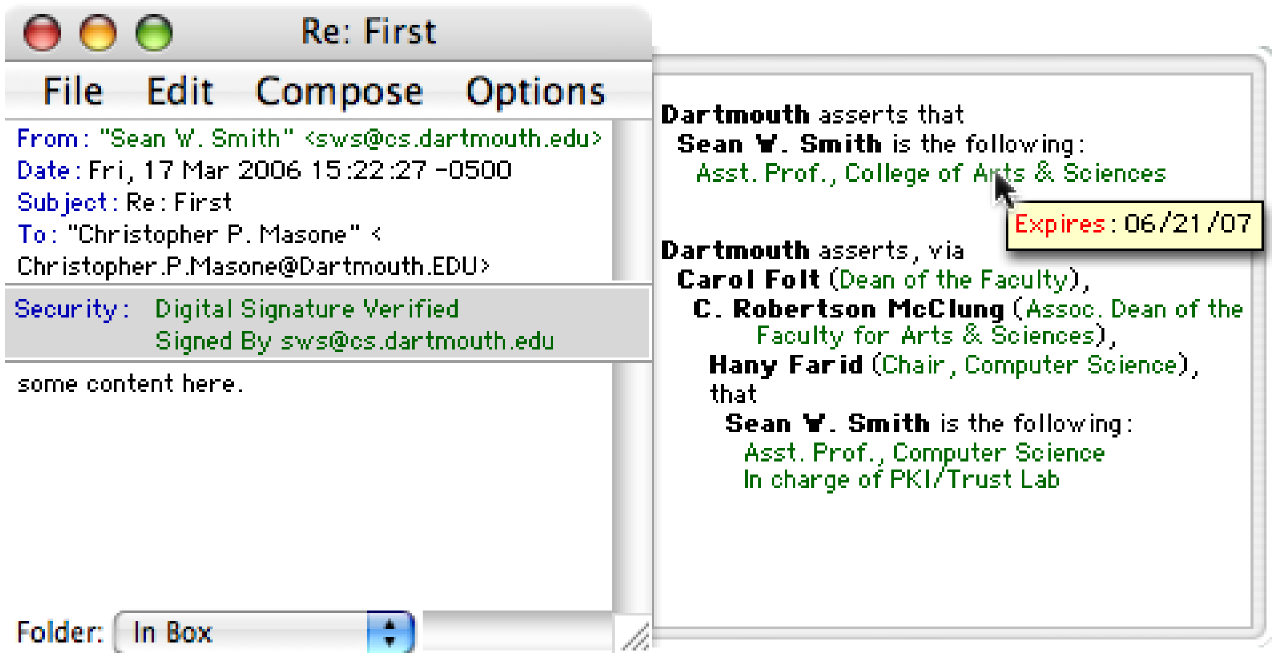
For the dual purposes of demonstrating our design philosophy and helping users manage trust in secure email, we introduce the *Attribute-Based, Usefully Secure Email (ABUSE)* system. Rather than attempt to automatically make trust decisions for users, the system is designed to help them make more informed trust decisions about email that they receive. We do this by allowing users to create useful metadata about each other, access the store of data about themselves, and attach selected attributes to outgoing messages. Then we present this information to recipients in an understandable fashion. Our design goals are to

1) enable users to bind appropriate trustworthy assertions about themselves to outgoing email,
2) enable users to understand trustworthy assertions about senders of incoming email,
3) avoid push-back from users without ABUSE-saavy clients,
4) minimize the administrative burden on everyone involved,
5) avoid the need for an organization-wide "Attribute Administrator",
6) avoid limiting the attribute space (i.e. avoid predefining a set of attributes and relationships),
7) leverage existing PKI and S/MIME infrastructure, and
8) provide some support for attributes belonging to users at outside organizations.

In the case of the email "from the Dean" discussed earlier, the Dean's assistant could have cryptographically bound an attribute—given to him by the Dean herself—to his message stating his relationship to her. Recipients would then have been able understand the situation without having to keep track of who works for the Dean.

A. *Creating, Storing, Distributing and Displaying Attributes*

In order to achieve the first three design goals above, we need to design and build ways to create, store, distribute and display ABUSE attributes without impacting users of standard email clients. As we will discuss in Section V-B, we envision ABUSE users creating attributes on their own, without having to involve a system administrator. Addressing this portion of the system, therefore, is basically a user-interface design problem, as is the attribute display portion. Prior work has been done on designing usable user interfaces for secure email [2], [3], which we



From: "Sean W. Smith" <sws@cs.dartmouth.edu>
Date: Fri, 17 Mar 2006 15:22:27 -0500
Subject: Re: First
To: "Christopher P. Masone" <
Christopher.P.Masone@Dartmouth.EDU>

Security: Digital Signature Verified
Signed By sws@cs.dartmouth.edu

some content here.

Dartmouth asserts that
Sean W. Smith is the following:
Asst. Prof., College of Arts & Sciences
Expires: 06/21/07

Dartmouth asserts, via
Carol Folt (Dean of the Faculty),
C. Robertson McClung (Assoc. Dean of the Faculty for Arts & Sciences),
Hany Farid (Chair, Computer Science),
that
Sean W. Smith is the following:
Asst. Prof., Computer Science
In charge of PKI/Trust Lab

Fig. 1. A mock-up of a GUI for presenting ABUSE attributes to users. This display is based on the message reading window for BlitzMail, Dartmouth's own email system. The new features include the pane labeled "Security", which allows users to see the verification status of the message along with the name of the signer. Note that, in this example, although the signer and the sender do not match, a user who knows "Sean W. Smith" will be aware that both addresses belong to the same person and thus be able to conclude that this mismatch is ok. The tray on the right, containing attribute information, is also a feature we added. The coloring of the attributes denotes their verification status, and we are using indentation to express the relationships between attributes in the same chain.

will discuss in Section VII. Additionally, there is a growing body of work on the general subject of usability in security (HCI-SEC), which has also informed our work. Garfinkel provides an excellent survey of the space in Chapter 2 of his PhD thesis [3], and also sets out several techniques for designing software to be both secure and usable. When designing the user interfaces for ABUSE, we will employ these strategies to ensure that users do not simply ignore the system, but instead actually use it to enhance their trust decisions. We have begun by using an iterative design approach [25], testing our prototype GUI for presenting ABUSE-enhanced messages to users (shown in Figure 1) in a pilot user study, which is discussed in Section V-F.

To facilitate future user studies, we are building ABUSE into Dartmouth's homegrown email client, known as BlitzMail. The vast majority of email usage at the college occurs through BlitzMail, and the users cross all demographics, from students to faculty to staff. We feel that the size and variety of this installed base will provide us with a good volume of data, and that the users' familiarity with the client will allow us to avoid worrying about users being confused with the general email portion of the UI when designing our studies. We believe the benefits of cleaner user studies are worth taking on the challenge of integrating ABUSE with BlitzMail.

In comparison to the iterative and subjective process of user interface design, creating an infrastructure for storing and distributing ABUSE attributes should be a much simpler task. Arguments could be made for storing attributes on the client-side, but some details of the client platform upon which we are building our prototype dictate that we should instead provide a central attribute store, indexed by users' public keys. A user's client will have to prove knowledge of their private key in order to pull attributes out of the central attribute directory. Then, the client will allow the user to choose which of their attributes they wish to attach to a given message, if any. The sender's private key is then used to sign a hash of the chosen attributes and the message, and this signature is included with the rest of the ABUSE content. To handle distribution of attributes without causing push-back from users without ABUSE-enabled clients, we will borrow an idea from Stream/CoPilot [3], and DKIM [26] and put attributes into messages as MIME or RFC 2822 headers. We tested Mozilla Thunderbird, Microsoft Outlook Express and Apple Mail to verify that these popular clients simply ignore headers that they do not understand, so users of non-ABUSE clients would likely not be bothered by communicating with parties who use the system. Forwarding behavior varied, with some clients stripping out headers for encapsulated messages and others maintaining them, but hiding them. We have not addressed the issue of attributes in forwarded mail, but as long as headers are maintained, this can be dealt with.

B. *Attribute Management*

Our goals of minimizing the administrative burden on users and avoiding the need for a dedicated administrator are somewhat in opposition. We hope to balance the two by allowing users to grant attributes to each other in a manner similar to SDSI/SPKI [27], [28] and the PERMIS project [29], in addition to the Greenpass [30] work done here in our lab. Unlike PERMIS, but like Greenpass, we use public keys to identify users, as opposed to X.509 Distinguished Names. Unlike Greenpass, but like PERMIS, we use chains of X.509 Proxy Certificates (PCs) [31] to store attribute information. PCs are structurally similar to X.509 Identity Certificates except that they tend to be more short-lived and include some extra extensions that allow for the specification of arbitrary information. Chains of PCs can be created just like it is possible to create chains of identity certificates. So, if Alice possess an ABUSE attribute A (a chain of PCs) and she wishes to grant to "Sean Smith" a new attribute that chains off of A, she must create a new chain of PCs B that consists of A with a single new PC tacked onto the end. To do this, she acquires his public key, which is available in Dartmouth's LDAP, issues a PC containing the specification of the attribute using his public key as the *subject* of the certificate, creates B by attaching this certificate to the end of A and inserts it into the ABUSE directory specified above. Attributes can be verified by recipients in the same way that identity certificate chains are verified.

We envision bootstrapping the attribute generation process by having a local trust-root grant a small set of attributes to some high-level members of the organization. In a university environment, perhaps this set would consist of the high-level administrators (Deans, the President and so forth). Then, they could each grant attributes to the people beneath them. For example, the Dean of the Faculty could grant "Chairperson" attributes to the chair of each academic department, who could in turn handle granting attributes conferring professorial status to members of their own department. In this way, responsibility for maintaining portions of the attribute space is divided among many people, but each individual is only responsible for a small portion of the overall space. Since attributes all chain back to a single trust root, as long as a user's client is configured to trust that root, the client will be able to verify that the attributes are valid. This is not unreasonable, since we are considering an environment inside an institution, where all clients can be configured to use a local Certification Authority (CA) as their trust root.

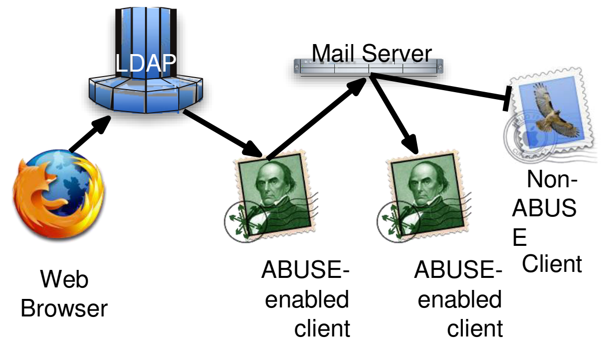


Fig. 2. The various components of a single enterprise ABUSE system. Users create attributes with a web browser and publish them to an LDAP. Using an ABUSE-enabled email client, senders of email can retrieve their attributes, attach them to outgoing email, and send them along. Recipients with an ABUSE-enabled client can view this attribute info in conjunction with the message, while non-savvy clients will simply ignore the extra information.

Since we are planning to enclose attributes in X.509 certificates, it is natural to wonder why we chose not to use X.509 Attribute Certificates (ACs) [32]. Unfortunately, ACs are not widely supported among cryptographic libraries, which would complicate implementation, development and delopyment. There is software available to add such support but, since PCs are sufficient for our needs, we decided to avoid the added complexity.

C. *Attribute Content*

Some TM languages can only express certain kinds of relationships between principals. We do not believe that we can anticipate every relationship that a user would want to express between herself and another user, so defining a set of legal relations would be undesirable. Other languages are sufficiently flexible, but imposing significant structure on attributes would either make the user interface for creating attributes more complex, or require the system to convert back and forth from an easily human-parsable format. Since we are not trying to automatically reason about attributes in ABUSE, this seems unnecessary.

D. *Leveraging Existing Infrastructure*

Our seventh goal, leveraging existing PKI and S/MIME infrastructure is essentially a software engineering issue which led us to choose Proxy Certificates (PCs) to represent ABUSE attributes. While several TM systems can work with standard X.509-based certificates, they all require some software to convert the credentials to an internal representation. Since our email client will already include libraries to process X.509 certificates, adding both conversion code and libraries to deal with the new credential format seems extraneous. This same logic eliminated SDSI/SPKI, SAML [33], XACML [34], and XrML [35] from consideration for use in ABUSE; it is likely that these technologies would work, but the extra software engineering issues outweigh potential benefits.

E. Attributes From Outside Organizations

Our final goal of supporting attributes belonging to users from outside the organization is also the least integrated. We have focused first on getting ABUSE to work in a single institution. Our plans for scaling ABUSE beyond that limitation will be addressed in Section VI. Figure 2 shows the envisioned architecture of a single-enterprise ABUSE system, while Figure 3 shows an ABUSE deployment that spans multiple organizations.

F. A Pilot User Study for ABUSE-Enhanced Messages

Our pilot user study, which we performed only on the GUI for presenting ABUSE-enhanced messages to user, was modeled after the first round of testing related in [36]. We recruited five Dartmouth students who knew nothing about ABUSE, nor had any special knowledge of computer security, three undergraduates (a Biology major, a Government major, and a Sociology major) and two Computer Science graduate students (researching communication complexity and sensor networks). After taking a short survey regarding their attitudes towards computers, online privacy and security in general, the subjects were briefed about the forgeability of standard email (source addresses can be faked, content can be changed in transit, etc.) and informed of the existence of methods for verifying the sender of email. The participants were then exposed to several messages, some meant to look forged, some ABUSE-enhanced to make them appear more legitimate, and some with ABUSE attributes meant to show that they could *not* be trusted. In each case, the subject was asked by the message to perform some “dangerous” action: download and run an application, give up private information, or allow a student to skip an assignment. The subjects were given as much time as they wanted to look at each message. When ready, they had to choose whether to perform the requested action, and give a reason for that decision. Their choices and answers were anonymously recorded by the web application used for the test.

The study met with mixed success. One set of conditions wound up essentially testing whether the students felt that an intramural soccer game was a valid excuse for an extension, which was not our goal. Another discovered that our two graduate students are paranoid, and claim they will never download and open an attachment from email. However, in general, the subjects indicated a greater inclination to trust messages with ABUSE attributes that lent credence to the body of the message. For instance, in the condition where they were shown an ABUSE-enhanced message from an administrative assistant in the office of their employer asking for some personal information, they were willing to respond with the data. When they were shown a plain MIME email asking for that data, or an ABUSE-enhanced message with attributes showing the sender to be just a fellow employee, most declined to give up the information.

In general, in post-study interviews, the subjects felt the extra information helped, though one indicated that the information conveyed by ABUSE could usually be found online or by some other channel. Another agreed, but stated that she would be unlikely to actually check those sources in real life, so having it right there next to the message was useful. Some subjects indicated that they would like to be able to get more information about how the system worked, because the indenting scheme didn’t really convey the idea that attributes are chained. Rather they thought that a group of individuals had all attested to the final attribute in the chain. The interface currently only uses mouseover boxes to post expiration dates for the ABUSE attributes. In our next design we will add further information to the mouseovers and consider representing the attributes as a graph to better express how they relate to each other. Also, we will redesign the study conditions so that we better isolate usability information (instead of inadvertently testing users’ beliefs about the relative import of student activities).

The GUI tested is currently in the mockup phase. We understand from some of our sociologist colleagues that users are less likely to suppress criticism if they perceive the interface as a prototype, so we expect to do several more cycles of testing in this way. We have set up the ABUSE credential store, however, and are currently building the back end for delegating credentials from one user to another.

VI. Future Work

There are two portions of this project that are still pending: evaluation of a single-institution ABUSE system, and expanding ABUSE beyond the borders of one

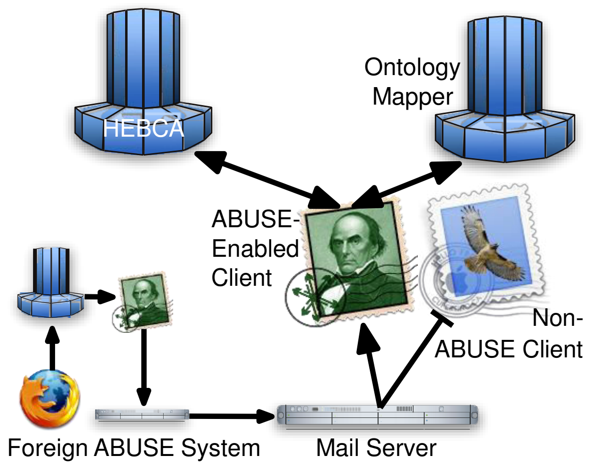


Fig. 3. An ABUSE system that spans multiple enterprises. As in the simpler setup, users create attributes with a web browser and publish them to an LDAP. Using an ABUSE-enabled email client, senders of email can retrieve their attributes, attach them to outgoing email, and send them along. ABUSE-enabled clients at external organizations that read this mail will use HEBCA to help validate the incoming attributes, use an ontology mapping service to help provide some local context for these foreign attributes, and then present this information to the user, along with the message. Again, non-savvy clients will simply ignore the extra information.

organization.

A. Evaluating ABUSE

In addition to making heavy use of user studies during the user interface design portions of building ABUSE, we also plan to deploy our prototype to a large community of users and collect feedback on their usage patterns. As mentioned earlier, we are building ABUSE on top of BlitzMail, due to its prevalence at the College. We believe this will both allow us to conduct broader user studies, as well as providing us with clearer results, due to user familiarity with the basic user interface.

B. ABUSE Across Organizations

To take ABUSE beyond a single institution, we must address both the issue of verifying attribute chains from foreign sources and also that of mapping unfamiliar attributes into a locally sensical context. Bridge Certificate Authorities [37] provide a method of joining disparate hierarchical X.509-based PKIs in a non-hierarchical way. By making both ABUSE attributes and S/MIME digital signatures on ABUSE-enabled emails "bridge aware", we can not only address the problem of verifying foreign attributes, but also begin exploring the possibilities and pitfalls of bridged PKIs. The Higher Education Bridge Certification Authority (HEBCA) [38] has been set up and deployed at Dartmouth, so we should be able to evaluate our bridged applications as they work with real, deployed infrastructure.

To help users make sense of attributes created at outside institutions, we plan to apply some ontology mapping results from the W3C's Semantic Web project [39]. Ontology mapping uses machine learning techniques to attempt to map one hierarchical classification structure onto another [40], [41]. An organization's attribute space can be viewed as an ontology, since it is structured like a tree, rooted at the local CA. We believe that an ontology mapper could be trained on this "home ontology" , and then treat incoming sets of attributes as portions of a foreign ontology and attempt to perform a mapping. While it is unlikely that this will provide a perfect solution, we hope to be able to provide some kind of confidence measure along with the mappings that we perform.

VII. Related Work

In addition to the TM work discussed at length above, there are two other groups of related work: systems directly related to trust in email, and general work on usability in secure email.

A. Trust in Email

Both S/MIME, upon which ABUSE is built, and PGP/MIME [42] can be considered work in this space. Digital signatures can, in many cases, provide users with enough context to decide whether or not to trust an incoming message. However, as we have discussed, there are cases that are not addressed by S/MIME and work done by other researchers [1] has shown that PGP/MIME clients are not usable by average users. To our knowledge, only Role-Based Messaging [43], [44] has attempted to address the same portion of the problem space as our work.

Role Based Messaging (RBM) is a system that creates role-based mail accounts. Users who have appropriate credentials (where "appropriate" is defined by policy on a per-role basis) can log into those accounts to read mail sent *to* that role and also to send signed and encrypted mail *from* that role. Mail may be encrypted to a role, not simply to a specific user. Role membership is controlled by a PERMIS [29] back end, in which X.509 ACs are used to store role membership information. Policies can be added to messages to further control what recipients can do with them. A policy governs who can assign roles, though this could be set up to allow any user to grant roles to others. Also, these “role managers” *can* create new roles within their organization, but they will not be recognized by the system [45].

Recent RBM work has proposed Policy Based Management (PBM), which adds infrastructure to allow organizations to advertise what kinds of policy languages they support [46]. PBM also allows third parties to sign off on particular implementations of particular policy languages and enforcement models, so that an enterprise can prevent (again, via policy) users from sending secure messages to an external organization whose mail system may not respect message permissions set by the sender.

In addition to several other issues, RBM offers a solution for the email trust problem. Users could all be granted roles, and then choose the appropriate role from which to send a message that needed to be trusted. It does not appear that users could claim multiple roles at the same time, but new, combined roles could feasibly be created to handle that. Also, it is unclear whether the nice audit feature provided by the chaining of ACs to create ABUSE attributes is also provided by RBM. Most importantly, while the authors mention “user friendliness” as a design goal in one of their early papers, recent correspondence indicates that usability has been very much a second tier goal in their work thus far [47]. According to the authors, the have so far implemented an RBM policy decision engine and a distributed RSA algorithm that they plan to use to avoid having a single point of compromise in their system architecture. They plan to use Mozilla Thunderbird as a client platform for RBM, but did not say much about the RBM user experience beyond that.

We believe ABUSE’s focus on usability will make it a more usable, and therefore deployable, solution to the email trust problem than RBM.

B. *Secure Email Usability*

Both Garfinkel and Whitten have developed clients for secure email that focus on usability [2], [3]. Whitten’s ”Lime” system was designed to help users understand the key certification portion of a PGP/MIME email system. Garfinkel’s Stream and CoPilot systems were built to abstract signing and encryption away from users when emailing with parties with whom they communicate on a regular basis. Neither system focused on the problem of providing more context for users trying to make trust decisions regarding incoming messages.

VIII. Conclusion

In this paper, we have introduced Attribute-Based, Usefully Secure Email (ABUSE), which we are building to exemplify our principle of leveraging humans in the design of secure systems. We chose to address secure email in particular due to several concerns about the expressiveness of S/MIME email technology, including cases in which names lack specificity, properties—not names—influence trust decisions, and properties mean different things in different contexts. ABUSE addresses these first two concerns by enabling users to delegate trustworthy attributes to each other, and then bind them to S/MIME messages sent over email. Humans are leveraged at both ends of the process: humans hand out attributes to each other, and humans decide whether the attributes bound to a message are enough to build trust in the displayed content. The third concern is addressed by bridging across distinct PKIs and by mapping foreign attributes into a local context. Through development and testing of ABUSE, we hope to answer two long-term questions: whether issuing credentials in distributed way will actually work, and also whether users will actually understand these distributed credentials, enabling them to make more accurate trust judgments about incoming messages from unfamiliar senders.

References

[1] A. Whitten and J. Tygar, “Why Johnny Can’t Encrypt: A Usability Evaluation of PGP 5.0.” in *8th USENIX Security Symposium*, 1999.

[2] A. Whitten, “Making security usable,” Ph.D. dissertation, Carnegie Mellon University School of Computer Science, 2003.

[3] S. Garfinkel, “Design principles and patterns for computer systems that are simultaneously secure and usable,” Ph.D. dissertation, Massachusetts Institute of Technology, 2005.

[4] L. G. Zucker, “Production of trust: Institutional sources of economic structure, 1840–1920,” in *Research in Organizational Behavior*. JAI Press Inc., 1986, vol. 8, pp. 53–111.

[5] Y.-H. Chu, J. Feigenbaum, B. LaMacchia, P. Resnick, and M. Strauss, “REFEREE: Trust management for Web applications,” *Computer Networks and ISDN Systems*, vol. 29, no. 8–13, pp. 953–964, 1997.

[6] A. Schutz, “On multiple realities,” in *Collected papers 1: the problem of social reality*, M. Natanson, Ed. The Hague: Martinus Nijhoff, 1962, pp. 207–259.

[7] H. Garfinkel, “A conception of and experiments with “trust” as a condition of stable concerted actions,” in *Motivation and social interaction: Cognitive determinants*, O. Harvey, Ed. New York: Ronald Press, 1963, pp. 187–239.

[8] B. Ramsdell, “Secure/Multipurpose Internet Mail Extensions (S/MIME) version 3.1 message specification,” July 2004, RFC 3851.

[9] ——, “Secure/Multipurpose Internet Mail Extensions (S/MIME) version 3.1 certificate handling,” July 2004, RFC 3850.

[10] R. Housley, W. Ford, W. Polk, and D. Solo, “Internet X.509 Public Key Infrastructure Certificate and CRL Profile,” 1999, RFC 2459.

[11] D. R. Kuhn, V. C. Hu, W. T. Polk, and S.-J. Chang, "Introduction to public key technology and the federal PKI infrastructure," http://www.csrc.nist.gov/publications/nistpubs/800-32/sp800-32.pdf, NIST, February 2001.

[12] R. Nielsen, "Observations from the deployment of a large scale PKI," in *4th Annual PKI R&D Workshop*, C. Neuman, N. E. Hastings, and W. T. Polk, Eds. NIST, August 2005, pp. 159–165.

[13] A. Kapadia, "personal communication," aug. 29,, 2006.

[14] S. W. Smith, C. Masone, and S. Sinclair, "Expressing trust in distributed systems: the mismatch between tools and reality," in *Forty-Second Annual Allerton Conference on Privacy, Security and Trust*, September 2004, pp. 29–39.

[15] J. Beale, "personal communication," sept. 3, 2006.

[16] N. Li and J. C. Mitchell, "RT: A role-based trust-management framework," in *Proceedings of The Third DARPA Information Survivability Conference and Exposition (DISCEX III)*. IEEE Computer Society Press, Los Alamitos, California, April 2003, pp. 201–212.

[17] N. Li, B. N. Grosof, and J. Figenbaum, "Delegation logic: A logic-based approach to distributed authorization," *ACM Transactions on Information and System Security (TISSEC)*, vol. 6, no. 1, pp. 128–171, February 2003.

[18] N. Li, J. C. Mitchell, and W. H. Winsborough, "Design of a role-based trust management framework," in *Proceedings of the 2002 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, Los Alamitos, California, May 2002.

[19] ——, "Beyond proof-of-compliance: Security analysis in trust management," *Journal of the ACM*, vol. 52, no. 3, May 2005.

[20] T. Jim, "Sd3: A trust management system with certified evaluation," in *SP '01: Proceedings of the 2001 IEEE Symposium on Security and Privacy*. Washington, DC, USA: IEEE Computer Society, 2001, p. 106.

[21] A. Herzberg, Y. Mass, J. Michaeli, D. Naor, and Y. Ravid, "Access control meets public key infrastructure, or: Assigning roles to strangers," in *Proceedings of IEEE Symposium on Security and Privacy*, May 2000, pp. 2–14.

[22] M. Blaze, J. Feigenbaum, and J. Lacy, "Decentralized trust management," in *Proceedings of IEEE Symposium on Security and Privacy*, May 1996, pp. 164–173.

[23] M. Blaze, J. Figenbaum, J. Ioannidis, and A. D. Keromytis, "The KeyNote trust-management system version 2," September 1999, RFC 2704.

[24] H. Cunningham, "Information Extraction, Automatic," *Encyclopedia of Language and Linguistics, 2nd Edition*, 2005.

[25] D. A. Norman, *The Design of Everyday Things*. Basic Books, 1988.

[26] E. Allman, J. Callas, M. Delaney, M. Libbey, J. Fenton, and M. Thomas, "DomainKeys Identified Mail Signatures (DKIM)," April 2006, Internet Draft, http://www.ietf.org/internet-drafts/draft-ietf-dkim-base-01.txt.

[27] R. Rivest and B. Lampson, "SDSI - A Simple Distributed Security Infrastructure," April 1996, http://theory.lcs.mit.edu/ ˜ rivest/ sdsi10.html.

[28] C. Ellison, B. Frantz, B. Lampson, R. Rivest, B. Thomas, and T. Ylnen, "SPKI Certificate Theory," September 1999, RFC 2693.

[29] D. Chadwick, "The PERMIS X.509 role based privilege management infrastructure," in *Proceedings of 7th ACM Symposium on Access Control Models and Technologies (SACMAT 2002)*, 2002.

[30] N. Goffee, S. Kim, S. Smith, W. Taylor, M. Zhao, and J. Marchesini, "Greenpass: Decentralized, PKI-based Authorization for Wireless LANs," in *Proceedings of 3rd Annual PKI R&D Workshop*. NIST/NIH/Internet2, April 2004.

[31] S. Tuecke, V. Welch, D. Engert, L. Pearlman, and M. Thompson, "Internet X.509 Public Key Infrastructure (PKI) Proxy Certificate Profile," 2004, rFC 3820.

[32] S. Farrell and R. Housley, "An Internet Attribute Certificate Profile for Authorization," 2002, RFC 3281.

[33] S. Cantor, J. Kemp, R. Philpott, and E. Maler, "Assertions and protocols for the OASIS security assertion markup language (SAML) v2.0," 2005, http://docs.oasis-open.org/security/saml/v2.0/saml-core-2.0-os.pdf.

[34] T. Moses, "eXtensible Access Control Markup Language (XACML) version 2.0," 2005, http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-core-spec-os.pdf.

[35] "XrML frequently asked questions," http://www.xrml.org/faq.asp, visited on Sept. 30, 2006.

[36] S. Brostoff, M. A. Sasse, D. Chadwick, J. Cunningham, U. Mbanaso, and O. Otenko, "RBAC What? Development of a Role-Based Access Control Policy Writing Tool for E-Scientists," in *Workshop on Grid Security Practice and Experience, Oxford, UK*, July 2004, pp. V21–38. [Online]. Available: http://www.cs.kent.ac.uk/pubs/2004/2067

[37] R. Houskey and T. Polk, *Plannning for PKI*. Wiley, 2001.

[38] "Educause — educause major initiatives — higher education bridge certification authority," http://www.educause.edu/HEBCA/623, visited on Jan. 24, 2007.

[39] "W3C semantic web," http://www.w3.org/2001/sw/, visited on Sept. 30, 2006.

[40] P. Bouquet, L. Serafini, and A. Zanobini, "Semantic Coordination: A New Approach and an Application," in *2nd International Semantic Web Conference*, October 2003, pp. 20–23.

[41] A. Doan, J. Modhavan, P. Domingos, and A. Halevy, "Learning to Map Between Ontologies on the Semantic Web," in *Proceedings of The Eleventh International WWW Conference*, May 2002.

[42] M. Elkins, "MIME security with pretty good privacy (PGP)," October 1996, RFC 2015.

[43] D. Chadwick, G. Lunt, and G. Zhao, "Secure Role-based Messaging," in *Eighth IFIP TC-6 TC-11 Conference on Communications and Multimedia Security (CMS 2004),Windermere, UK*, unknown 2004. [Online]. Available: http://www.cs.kent.ac.uk/pubs/2004/2069

[44] G. Zhao and D. Chadwick, "Evolving messaging systems for secure role based messaging," in *10th IEEE International Conference on Engineering of Complex Computer Systems (ICECCS'05)*, June 2005, pp. 216–223. [Online]. Available: http://www.cs.kent.ac.uk/pubs/2005/2231

[45] G. Zhao, "Personal communication," apr. 12, 2006.

[46] G. Zhao and D. Chadwick, "Trust infrastructure for policy based messaging in open environments," in *14th IEEE International Workshops on Enabling Technologies: Infrastructures for Collaborative Enterprises (WETICE-2005)*, Linkping University, Sweden, June 2005. [Online]. Available: http://www.cs.kent.ac.uk/pubs/2005/2233

[47] G. Zhao, "Personal communication," apr. 11, 2006.