Expressing Trust in Distributed Systems: the Mismatch Between Tools and Reality[1]

Sean W. Smith Chris Masone Sara Sinclair
Department of Computer Science
Dartmouth College {sws,cmasone,sinclair}@cs.dartmouth.edu

Abstract. Distributed systems typically support processes that involve humans separated by space and by organizational boundaries. Because of its ability to enable secure communications between parties that do not share keys a priori, public key cryptography is a natural building block for the elements of these computing systems to establish trust with each other. However, if the trust structure we build into the computing systems does not match the trust structure in the human systems, then this trust infrastructure has not achieved its goal. In this paper, we assess the inability of the standard PKI-based tools to capture many trust situations that really arise in current distributed systems, based on our lab's experience trying to make these tools fit. We offer some observations for future work that may improve the situation.

1. Introduction

Public key cryptography offers a host of solutions to problems that arise from bringing a very human concept—trust—into the electronic world. These solutions in some cases allow us to accomplish things that we could not before. Public key cryptography can also help us establish a stronger bond of trust than previous methods of signing could; given a message signed by Alice and an identity cert issued by an authority that Bob trusts, he might even be more confident that she says who she says she is than if he received a paper letter or a phone call.

For these electronic trust systems to be truly useful to humans, however, we must make sure that they mirror and extend parallel systems in the real world. In other words, the systems we develop should not limit the ways in which we use digital trust; they should be understandable within the framework of our real-world trust models, and able to interface with those models when necessary.

As we will see, this ideal vision of electronic trust is not yet realized by current *public key infrastructure (PKI)* systems. We will thus look at ways in which current PKI falls short of this goal, and toward a solution for this problem.

This Paper Section 2 reviews the basic tools of public key cryptography and PKI. Section 3 presents work examining the mismatch between human users and these basic tools. Section 4 through Section 7 catalog instances where the tools fail to capture the appropriate nuances for trust judgment in distributed settings. Section 8 tries to distill this catalog into some necessary properties for the deployed trust tools to have, and concludes with examining some potential ways to achieve these properties in practice.

2. PKI Tools in Practice

Public key cryptography is designed to enable communication between parties who share no secrets a priori. This is achieved through each party having two keys: one

[1]This research has been supported in part by Intel, Cisco, and the Mellon Foundation, AT&T/Internet2 and the Office for Domestic Preparedness, Department of Homeland Security (2000-DT-CX-K001). This paper does not necessarily reflect the views of the sponsors.

private, which hopefully remains secret; and one public, which is published for others to use. It is thus possible for a party who knows Alice's public key to *encrypt* messages that only Alice can decrypt and read; for Alice to *sign* a message (by, for example, providing an encrypted hash of the message) that can be verified to have come from her; for a third party to be able to verify these signatures (*non-repudiation*); and for Alice to *authenticate* herself to a server, network, or other entity through a protocol that involves proving knowledge of her private key.

In order to make public key cryptography actually work in the real world, a certain amount of infrastructure surrounding the cryptographic operations is necessary. As generally interpreted, *public key infrastructure* focuses how to let Alice know what Bob's public key is. In the current basic approach, a *certification authority (CA)* signs an X.509v3 identity certificate that contains Bob's public key along with other important information about him, such as his name, his email address, and the organization to which he belongs. In practice, PKI discussion has been dominated by logistical questions. How does Alice obtain Bob's certificate once it has been issued? How does she decide to trust Bob's CA to make assertions about him? How does she check if Bob's CA has changed its mind?

Assessment It is very easy to focus on the technical end of PKI. However, a disconnect often exists between what works on paper and what human beings will understand, trust, and really use (and perhaps avoid misusing). We believe that such problems may have root in an underlying disconnect between what PKI systems *allow* users to do and what those users really *want* to do.

When assessing the effectiveness of these trust systems, we consider the following metrics. Does the trust structure of the machines match the trust structure of the human processes? Can humans accomplish what they want? Can they do so in a way that is parallel to relationships in the real world this distributed computation is supporting?

In our work with PKI systems, we have encountered several instances where the current PKI falls short of the stated goal. Ellison [Ell99] identified similar mismatches between human systems and PKI, and attributed them to erroneous assumptions made when PKI was first established; his solution involved systems that go beyond the "pure" X.509 PKI. Later work by Gutmann [Gut02] looks further into such problems.

3. Usability

Previous work has examined the mismatch between what human users and designers think the machines do, and what the machines really do.

Users and their Clients In 1999, Whitten and Tygar studied whether users could understand the PKI-based email tool that (at the time) was considered to have had the best user interface [WT99]. They found that as many users managed to send their secret message in plaintext, by accident, as managed to "successfully send encrypted email and decrypt a reply."

Such usability problems are considered by many to be inherent to security systems. However, follow-on work by Whitten [Whi] indicates that this may not necessarily be the case: her user studies showed that it was in fact possible to design a mail client with encryption and key management facilities that were understandable by the study participants. The application that she developed made use of "safe staging," where users are allowed to "safely and consciously postpone learning about the use of particular security mechanisms" until they are ready to do so.

Whitten and others (such as Garfinkel [Gar03]) further combat the association of security with lack of usability by offering concrete design principles that can guide the development of both usable and secure software. We can thus hope that current deficiencies in usability are not a property of security software, but instead a matter of design choices.

Designers and the Users and their Clients Our own lab has been exploring ways to use PKI to capture common trust expression in our university's electronic workflow processes, such as submitting homework, processing expense reports, and accessing university information services over the Web. However, we have repeatedly found that, in the standard tools, what the machines do with the PKI does not match what the humans using these PKI-based tools for workflow think the machines are doing. Furthermore, adversaries can exploit this mismatch.

A Web server can offer malicious content that a browser will render in a way mimics the signals a user expects for an SSL-secured Web site [YS02, FBDW97]; an adversary can construct Word documents and Excel spreadsheets whose apparent contents can change in usefully malicious ways without invalidating digital signatures on these files [KSA02, JDA02]; an adversary can use a number of techniques to produce Web requests SSL-authenticated with a client's private key, without that client's human being aware of these requests [MSZ04]. Nonetheless, these techniques—server-side SSL, client-side SSL, digital signature packages for Office and mail—are being used as the foundation for securing electronic manifestations of human processes.

4. Expressiveness: Name $\neq$ Person

The above discussion focused on *usability:* whether humans have the correct perceptions of what the machines are doing with these PKI-based trust computations. However, we've also repeatedly run into problems with *expressiveness:* these signed digital assertions, and the way our machine use them, do not let us express the right properties necessary for trust conclusions in human settings.

As an initial scenario, let's consider using S/MIME signatures on email. In the human setting, suppose that a message is allegedly from a specific human colleague; we'd like to use a digital signature to confirm that. An initial set of problems arise because the standard tools focus on the *name* of the alleged sender, but we want to know whether the sender is a particular *person.*

We enumerate some trouble areas.

One Name, Many People An initial problem is that one name may correspond to a set of people, even within the relatively limited population of a large enterprise. Carl Ellison repeatedly discusses what he calls the "John Wilson" problem: his former enterprise employed several people named "John Wilson," and often data—such as airline tickets, or perhaps even encrypted email traffic—intended for one would go to the other [Ell02]. The first author recalled a period in grad school at CMU when all email to anyone named "Smith" was routed to him, since "smith" was his userid. Within Dartmouth Computer Science, we have problems finding the email address[2] of "Joan Wilson," our grant manager, since the Dartmouth Name Directory lists many Joan Wilsons, distinguishable only by middle initial (not helpful) and department (also not helpful, because she is in a general administrative unit, not ours).

One Person, Many Names We can also see the flip-side of the above problem: one person can map to several different names. For example, a former researcher in our lab

[2]To protect the innocent, we have changed names to some variant of "John Wilson."

has appeared on paper by-lines as "Zishuang Eileen Ye," "Eileen Zishuang Ye," "Eileen Ye," and "Zishuang Ye," due to her co-author's misunderstanding of the Chinese student practice of adopting a Western nickname. Future scholars searching literature databases may wonder at the number of similarly-named researchers we had working on the same topic.

One Person, Many Accounts In the S/MIME mail idiom as practiced, a sender is identified by his certificate and email address. In mail as practiced at Dartmouth and elsewhere, a human user may have several email addresses—furthermore, which one appears as the sender address in the message may depend not just on the sender's choice of account, but on how that particular installation of that mail client chooses to do things.

For example, a colleague "John Wilson" sends email both as `John.Wilson@dartmouth.edu` as well as `jwilson@ists.dartmouth.edu`. John initially had trouble because the Dartmouth CA (which our lab runs) did not list the `ists` address in his certificate. Users of some S/MIME clients, such as Apple Mail, were thus not able to verify John's signatures—because the client decided that certificate for `John.Wilson` could not possibly be correct for a message from `jwilson`, and so looked no further.

One workaround for such a problem is for John to set himself up with multiple certified key pairs, one for each address. However, a problem here is how John can be sure to pick the right one for the right account. We've noticed that some platforms will list a choice of user key pairs when one is needed (although sometimes they each are identified by same name string, "Sean W. Smith"). The current Apple design appears to assume that a user will only ever have one key pair; should the user install more than personal key pair in his keystore, the platform silently picks one of them to use on its own.

In an extreme case of this problem, we noticed the Apple Mail client consistently failed to verify the signatures of an external colleague we'll call `John.Wilson@foo.com`. Investigation revealed that this failure stemmed from the fact that the message was from `John.Wilson@foo.com` but John's certificate said `john.wilson@foo.com`. Further investigation revealed that the Apple Mail client was actually correct—the standard says that everything before the "@" must match exactly. However, John could not fix this problem: his company's mailer applied the capitalization on its own, no matter how John configured his client; furthermore, when John requested a new certificate with the matching capitalization, the CA refused—since in the CA's view, capitalization didn't matter, and policy prohibited issuing duplicate certificates for the same address.

5. Expressiveness: Name $\neq$ Property

In the human setting, a recipient wants to give credence to a message not because of the person who sent it, but because of some particular role or property they have. The *name* of the sender is not the important aspect.

We enumerate some trouble areas.

Inferring Property from Name One immediate challenge is trying to infer a relatively short-lived property from a longer-lived identify certificate. Is "John Wilson" the Dean right now? Is co-author "Sara Sinclair" currently a full-time student? Can Sara convince her mother's health-insurance company (who wants to know, in order to sell her insurance) or Amazon (who wants to know, in order to give her a discount)?

The first author had a colleague in grad school whose first name was "Dean" (let's say "Dean Wilson"). Dean usefully exploited the fact that campus bureaucrats would interpret his name as his title—and found that having his female officemate call "on behalf of Dean Wilson" was even more effective.

The Eye of the Beholder An often overlooked aspect to this problem is that, sometimes, the sender's certifier may not necessarily know whether the sender has this property. Sometimes the sender herself doesn't know; the property may be a subjective function of the relying party's current belief system [Smi04]. For example, in the currently proposed TCPA/TCG software attestation scheme (or the earlier PKI-based scheme in the IBM 4758), a remote machine may try to convince the relying party of its trustworthiness by reporting an exhaustive description of the software configuration. This approach is somewhat akin to a police officer proving trustworthiness by providing a transcription of his DNA rather than exhibiting a badge.

Many People, Same Property Another trouble area is the fact that the property in question may be shared by several people, in ways not easily expressed by long-lived names. For example, a senior colleague has preached that we need academic PKI, to prevent a repeat of an incident at Yale in which someone forged email from the Dean canceling classes. However, here at Dartmouth, we receive messages from the Dean all the time—except they're never sent by the Dean, but instead by an administrative assistant. A PKI email system that tells the recipient that "Harry Bovik" really sent this message from "Dean John Wilson" won't help unless it can also tell the recipient that "Harry Bovik" is the new assistant to Dean Wilson, and speaks for him on such matters.

Temporal Issues An overarching issue in current PKI is how to determine whether a signed binding of name to key is still valid when the relying party needs to make a trust decision. However, human scenarios give rise to some subtle nuances that a simple look at expiration or revocation of the key-property binding captures. For a hypothetical example, if "John Wilson" stops being Dean, the relying party might want to know if John signed a message when he still was Dean, even though the binding has now expired. On the other hand, messages signed *after a binding has expired* can sometimes still carry meaning from that binding. For a real example, in our department, Doug McIlroy's current statements about operating systems are well-regarded because, decades ago, he had been manager of the group that created UNIX. Drawing conclusions about messages signed *before a binding starts* can also be useful. When the first author worked at IBM but had accepted a teaching job at Dartmouth, he had trouble convincing a publisher to send him an answer key for the textbook he would be using after his appointment formally began.

6. Property $\neq$ Property

In the human setting, a recipient may also want to make a decision based on a property about a sender from a different enterprise. Even if we assumed enterprises had schemes in place to bind properties and roles to users, how properties at a remote enterprise should map to properties at the relying party's institution is not at all clear.

For example, it would be nice to process graduate program applications electronically. However, knowing that "John Wilson" digitally signed the transcript from an applicant from University X won't help unless we also know that John Wilson can speak for the equivalent at University X of what Dartmouth calls the "Office of the Registrar." How do we do this mapping?

Even in the academic domain, the situation can give rise to some surprising problems. For example, at most universities, the "Dean's List" designation indicates good academic performance; at Princeton, it was reputed to be the list of students on disciplinary probation—non-Princeton relying parties will map the property to its opposite. For another example, at Dartmouth, the "Coach" role implies that the holder is a responsible adult whose judgment deserves respect. When a colleague who moved to a different university had a student ask for an extension due to an event involving an athletic team that student was on, our colleague decided to grant it, but only if the team's coach said it was all right. "John Wilson" then sent mail saying it was all right. "John Wilson" really was the coach. "John Wilson" was also a student, happy to help his friend get out of classwork.

7. Delegation

In the human setting, the holder of a property often wants to share or delegate it to another party. However, in the common PKI structure, a property is typically something that is bound to one party somewhat permanently, and the binding must be performed by a special authority.

We enumerate some trouble areas.

Subcontracting PKI tools that focus on permanent names do not express subcontracting relationships. In a well-known example, if a user with a secured SSL connection to `palmstore` investigated the server certificate, he learned that he had a secure connection to "Modus Media." It turns out that Palm had subcontracted that part of their commerce service to Modus Media—but the name-centric SSL PKI had no way of telling the user that. For another example, our department has recently hired `linklings.com` to handle the solicitation and processing of letters of recommendation for graduate applications. In testing, a potential recommender received email from "Dartmouth Computer Science 2005 Graduate Submission and Reviews `<john@linklings.com>`." How can S/MIME tell this relying party that `john@linklings.com` really is authorized to act for Dartmouth? How can SSL tell the relying party that `https://secure.linklings.net/applications/dartmouth/cs2005/` really is the Dartmouth Computer Science application site? (We've produced forged SSL Web sites for class projects that look more convincing!) Should `linklings.com` obtain mail and SSL key pairs for each university they act for? Should they just obtain multiple certificates for their one key pair? If the latter, how do they know which certificate to present to the client?

Less Formal Authorization Current PKI tools also don't work well with more ad hoc relationships. For one example, suppose Alice at Dartmouth wanted to make some security materials available to students in a class a colleague Bob was teaching at University X. The real-world trust relationship goes through Bob: Alice wants her Web site to give materials to someone from X if Bob approves that. Alice shouldn't be bothered would enrolling the names of each student in this class; Bob shouldn't be bothered with becoming a formal X.509 Attribute Authority.

Similarly, what if Professor Alice wants to delegate the right to modify grades to Carlo the TA? (A colleague teaching a security course at a major university ended up needing to share passwords, because that was the easiest way to solve the problem in that university's infrastructure.)

In a related project, we have been looking at how to use PKI to secure access to a WLAN—while also allowing insiders to grant inside access to guests. Guest Bob doesn't need to visit the University President to get approval to come in the Computer Science building; it suffices that Professor Alice "said it was OK." However, EAP-TLS would require Bob to obtain a formal X.509 identity certificate from the University CA. (In our "Greenpass" work, we use a SPKI/SDSI hybrid PKI on top of X.509 to make this happen—without changing client software [GKS+04].)

8. Towards a Solution

From the above discussion, we can identify several aspects in which the expressiveness of deployed trust tools do not match human trust scenarios. We need to be able to express equivalence between various names and email accounts. We need to be able to express non-identity attributes, including wrinkles such as "speaks for" and "is going to be." We need to be able to express attributes across organizational boundaries. We need to be able to express delegation, without requiring heavyweight a priori relationships.

We might consider several approaches to overcome these problems.

X.509 Identity Certificates X.509 Identity Certificates are defined to be both extensible and flexible [HFPS99]. However, in its attempts to be good at everything, the standard prevents itself from being ideal for any one thing [Gut02]. For instance, colleagues of the authors here in the Dartmouth PKI Lab have suggested using the X.509 identity certificate SubjectAltName field as a place to store alternative names and email addresses to alleviate the one-name $\neq$ one-email-address problem we discussed earlier. However, several practical problems immediately present themselves. First of all, it is unclear if any mail clients actually check the SubjectAltName field while trying to do signature verification. Even if an X.509 ID cert issued to "Zishuang.Ye@dartmouth.edu" had "Zishuang.Eileen.Ye@dartmouth.edu" as an alternative name, would a mail client use it to verify a signature on mail from the "Zishuang.Ye" address? Furthermore, what if she then decided to just use "Eileen.Ye@dartmouth.edu" for simplicity? The CA would need to issue Eileen a whole new certificate, when all that she really wanted to do was add one string to the value of a field. A similar problem arises if she gets another email account, like in the "jwilson@ists.dartmouth.edu" case.

Attribute Certificates X.509 Identity Certificates are, by design, meant to be fairly permanent. Ideally, if one's key pair is never compromised, one could keep an ID cert forever. As we have shown in this paper, there are a wealth of attributes associated with an individual that should be accessible in a PKI, but many of those attributes are more transient than an ID cert. The PKI community realized this, and as a response created *Attribute Certificates (ACs)* [FH02]. ACs are meant to be shorter-lived than ID certs, and are normally created by an *Attribute Authority (AA)* to bind attributes to *Distinguished Names (DNs)* as used in ID certs. However, being an X.509 specification and based on ASN.1, ACs can be bent in various ways. For instance, instead of binding attributes to a DN, an AC can be created so that its attributes are instead bound to a public key. Also, as we have discovered in our exploration of PERMIS [Cha02, Cha04], some tools do not strictly require a distinguished AA; any entity in a PKI can create and sign ACs for any other entity. Chains of ACs could thus be created, with entity A making an AC for entity B, and then B turning around and making another AC for the same attribute for entity C. The Attribute Certificate RFC [FH02] explicitly recommends against this usage, however.

Even if we assume that chaining is impractical, ACs can still be useful. The obvious advantage is that sysadmins do not need to track of who has delegated which rights to whom. A second advantage has to do with the privacy of end-users. Assume that "Sara Sinclair" has both the attribute "Customer of Careful Car Insurance Co." and the attribute "Founding Member of the We-Crash-Lots Street Racing Club". It is likely that she would like to keep the latter attribute secret from her car insurance company, but would still wish to be able to communicate electronically and securely with them. From a more corporate point of view, having attributes about an employee's function in the same cert that conveys her public key gives away what could be considered proprietary data. Many organizations would prefer to keep information about their structure on a "need-to-know" basis. Using ACs could keep it that way. (Ongoing work in cryptographic approaches takes this privacy even further [Bra00, for example].)

SPKI/SDSI Close followers of PKI research may have noticed that the discussion of ACs above makes them sound like certificates from another, less traditional PKI variant. Using public keys to identify principals and allowing any entity to delegate attributes to any other entity are two of the main principles behind SPKI/SDSI [RL96, EFL$^+$]. A potential advantage of SPKI/SDSI over X.509 ACs is that an algorithm exists to discover certificate chains of SPKI/SDSI certs [CEE$^+$01] while, as stated above, no such algorithm is recognized for ACs. Some colleagues also insist that the lighter-weight syntax of SPKI/SDSI certs could make them easier to process and code for.

Proxy Certificates *Proxy Certificates (PCs)* are another X.509 variant designed to meet certain needs that are not handled by ID certs [TWE$^+$04, WFK$^+$04]. Initially designed to allow users to delegate some of their rights to processes running on their behalf, PCs are meant to be very short-lived (life of the process or less), bestow only the rights that they absolutely need to, and limit the damage in case the key pair associated with the PC is compromised. They accomplish this last feat by requiring that each PC be generated with a new key pair, thus assuring that an adversary that acquires the key pair can never do anything that the PC did not specifically allow. Moreover, these limited privileges die with the short-lived PC.

Bending the PC rules would obviously give us some more functionality. For instance, allowing them to be used to delegate to an existing key pair would allow them to be used more like ACs. This is a potential area for future work.

Hybrid PKI A "hybrid PKI" is an infrastructure that combines a traditional X.509-based PKI with some of the alternatives that we have discussed here. This approach leverages the fact that many entities already have X.509 ID certs that provide them with a key pair. For example, we can use the entity's public key as a "hook" onto which we hang ACs or PCs or SPKI/SDSI cert chains, or whatever other technology seems promising.

As mentioned earlier, Dartmouth's Greenpass project uses a hybrid PKI, attaching SPKI/SDSI certificate chains to a user's public key to control access to a secured wireless network [GKS$^+$04]. The underlying idea is that guests should not have to visit the University CA to get temporary access to our EAP-TLS secured wireless network. They should be able to get on because their hosts "said it was OK." Greenpass allows a guest and his host to sit down together and, via a Web interface, make this happen.

Cross-Organization Attribute Mapping Sending a user's attributes along with her digitally signed email would likely make it easier for the receiving party to make trust decisions about the message and/or the signer. However, things get more complex when messages are crossing organizational boundaries. If Alice, the "Head of the Office of Diversity" at University Y, receives a message signed by Bob, Dartmouth's "Associate Dean of Pluralism and Leadership", and they have no prior relationship, should Alice pay the message any heed? She might, if she were aware that the two positions are identical in all but name. It is obviously impractical to assume that every organization could keep a constantly updated mapping between every attribute held by every local user and every attribute held by every user at every other organization. Our current thoughts borrow some ideas related to the W3C Semantic Web [W3C]. The idea behind the Semantic Web is to allow information to be shared across enterprises without requiring everyone to reformat all their data. The CTXMATCH [BSZ03] algorithm, which uses lexical information to assist in the classification of outside data, and the GLUE system [DMDH02], which uses machine learning to find the closest match for new data in a “home” ontology are both potential starting points for attacking this problem. Obviously there is work to be done here as well. Combining an experiment here with the *Higher Education Bridge CA (HEBCA)*, a service our lab is hosting to provide basic inter-institution trust mapping, could be interesting.

Other Languages In addition to the X.509 variants and SPKI/SDSI, several other XML-based languages are currently being used to assert attributes and policies in a trusted manner across the web. For example, the *Secure Attribute Markup Language (SAML)* [Hal02] is used to securely assert user attributes across domains in *Shibboleth* [EC02], while the *eXtensible Access Control Markup Language (XACML)* is being created to allow users to define access control policies that use SAML statements as a source of information. These could play a role in some sort of hybrid PKI, though it is possible that the heavy-weight nature of XML could make this less practical.

9. Conclusion

In the real world, trust seems to be established on a local basis. If Sara and Chris grew up near one another and have a large number of mutual friends, they are more likely to trust one another than if they grew up on separate sides of the country. (Sociologists discuss theories of “social capital” to model this [Ant04].) Efforts to establish a hierarchy of CAs and certificates may work well in more formal social structures, such as the military, but do not scale well to the general population; instead of one or two roots CAs, we find ourselves with an increasing number and decreasing reason to trust them. However, we hope that this trend will not turn into a “web of trust” in which reputation deeply sufficient, for such systems tend toward having everyone trust everyone.

Thus, ideal trust schemes, if viewed as trees, should tend to be broad but not very deep. They should be flexible in allowing edges to be established or removed, as well as varied in the meanings of edges. They should not require all action to come from the top down, but they should allow for limited scope of power.

PKI as it is currently implemented does not reflect this type of structure, nor the more complex aspects of human trust discussed in this paper. It is still unclear whether the standards that define pure PKI even provide the means to achieve it, or if hybrid systems would be necessary. It is clear, however, while the disconnect between what humans do and what machines allow exists, PKI will remain hard for the former category of user to implement in their daily lives.

References

[Ant04] D. Anthony, 2004.

[Bra00] S. Brands. *Rethinking Public Key Infrastructures and Digital Certificates; Building in Privacy.* The MIT Press, 2000.

[BSZ03] Paolo Bouquet, Luciano Serafini, and Atefano Zanobini. Semantic Coordination: A New Approach and an Application. In *2nd International Semantic Web Conference*, pages 20–23, October 2003.

[CEE+01] D. Clark, J. Elien, C. Ellison, M. Fredette, A. Morcos, and R. Rivest. Certificate Chain Discovery in SPKI/SDSI. *Journal of Computer Security*, 9(4):285–322, 2001.

[Cha02] D. Chadwick. The PERMIS X.509 Role Based Privilege Management Infrastructure. In *7th ACM Symposium on Access Control Models and Technologies (SACMAT 2002)*, 2002.

[Cha04] D. Chadwick, September 2004.

[DMDH02] AnHai Doan, Jayant Modhavan, Pedro Domingos, and Alon Halevy. Learning to Map Between Ontologies on the Semantic Web. In *The Eleventh International WWW Conference*, May 2002.

[EC02] M. Erdos and S. Cantor. Shibboleth Architecture Draft v05. `http://middleware.internet2.edu/shibboleth/docs/draft-internet2-shibbol%eth-arch-v05.pdf`, May 2002.

[EFL$^+$] C. Ellison, B. Frantz, B. Lampson, R. Rivest, B. Thomas, and T. Ylnen. SPKI Certificate Theory. RFC 2693.

[Ell99] C. Ellison. The Nature of a Useable PKI. *Computer Networks*, 31:823–830, 1999.

[Ell02] C. Ellison. Improvements on conventional PKI wisdom. In *1st Annual PKI Research Workshop*. NIST/NIH/Internet2, April 2002.

[FBDW97] E. Felten, D. Balfanz, D. Dean, and D. Wallach. Web Spoofing: An Internet Con Game. In *20th National Information Systems Security Conference*, 1997.

[FH02] S. Farrell and R. Housley. An Internet Attribute Certificate Profile for Authorization. RFC 3281, 2002.

[Gar03] S. Garfinkel. Usable security: Design principles for creating systems that are simultaneously usable and secure, September 2003. PhD Thesis Proposal, MIT Department of Electrical Engineering and Computer Science.

[GKS$^+$04] N. Goffee, S.H. Kim, S.W. Smith, W. Taylor, M. Zhao, and J. Marchesini. Greenpass: Decentralized, PKI-based Authorization for Wireless LANs. In *3rd Annual PKI R&D Workshop*. NIST/NIH/Internet2, April 2004.

[Gut02] P. Gutmann. PKI: It's Not Dead, It's Just Resting. *IEEE Computer*, 35(8):41–49, 2002.

[Hal02] SAML v1.0, Assertions and Protocol. `http://www.oasis-open.org/committees/download.php/1371/`, 2002.

[HFPS99] R. Housley, W. Ford, W. Polk, and D. Solo. Internet X.509 Public Key Infrastructure Certificate and CRL Profile. RFC 2459, 1999.

[JDA02] A. Jøsang, D.Povey, and A.Ho. What You See is Not Always What You Sign. In *Proceedings of AUUG2002*, September 2002.

[KSA02] K. Kain, S.W. Smith, and R. Asokan. Digital Signatures and Electronic Documents: A Cautionary Tale. In *Advanced Communications and Multimedia Security*, pages 293–307. Kluwer Academic Publishers, September 2002.

[MSZ04] J. Marchesini, S.W. Smith, and M. Zhao. Keyjacking: the Surprising Insecurity of Client-side SSL. *Computers and Security*, 2004. To appear.

[RL96] R. Rivest and B. Lampson. SDSI - A Simple Distributed Security Infrastructure. `http://theory.lcs.mit.edu/~rivest/sdsi10.html`, April 1996.

[Smi04] S.W. Smith. Outbound Authentication for Programmable Secure Coprocessors. *International Journal of Information Security*, 2004. A preliminary version appeared in ESORICS 2002.

[TWE$^+$04] S. Tuecke, V. Welch, D. Engert, L. Pearlman, and M. Thompson. Internet X.509 Public Key Infrastructure (PKI) Proxy Certificate Profile. RFC 3820, 2004.

[W3C] W3C semantic web. `http://www.w3.org/2001/sw/`.

[WFK$^+$04] V. Welch, I. Foster, C. Kesselman, O. Mulmo, L. Pearlman, S. Tuecke, J. Gawor, S. Meder, and F. Siebenlist. X.509 Proxy Certificates for Dynamic Delegation. In *3rd Annual PKI R&D Workshop*, pages 31–47. NIST/Internet2/NIH, 2004.

[Whi] A. Whitten. *Making Security Usable*. Unpublished Ph.D. thesis, CS, CMU.

[WT99] A. Whitten and J.D. Tygar. Why Johnny Can't Encrypt: A Usability Evaluation of PGP 5.0. In *8th USENIX Security Symposium*, 1999.

[YS02] E. Ye and S.W. Smith. Trusted Paths for Browsers. In *11th USENIX Security Symposium*, August 2002. `http://www.cs.dartmouth.edu/~sws/papers/usenix02.pdf`.