Improvements on Conventional PKI Wisdom

Carl Ellison

carl.m.ellison@intel.com

Intel Labs

Abstract: *This paper contrasts the use of an ID PKI (Public Key Infrastructure) with the use of delegatable, direct authorization. It first addresses some commonly held beliefs about an ID PKI – that you need a good ID certificate to use digital signatures, that the ID certificate should come from a CA that has especially good private key security, that use of the ID certificate allows you to know with whom you're transacting and that the combination gives you non-repudiation. It then identifies flaws in those assumptions and addresses, instead, the process of achieving access control – either through an ACL plus ID, or directly. It then applies each method of achieving access control to two examples – one within a large company and one between companies.*

[This paper is an expanded transcript of the invited talk of the same title prepared for the Internet-2 1st Annual PKI Workshop, which was held at NIST at the end of April 2002.]

1 Introduction

The thesis of this paper is that the PKI community has accepted a number of concepts, listed here as "Conventional PKI Wisdom" that actually get in the way of achieving security. Some of them are false premises. Some of them are not achievable. None of them is necessary to achieve actual security. Instead, it advocates paying attention to the problem of access control and especially the determination of authorization. Authorization usually requires the same level of effort as ID certification. It can be used alongside ID certification, incurring extra load and expense, or it can be used instead of ID certification.

2 History

The concepts at issue here date back to the introduction of public key cryptography by Diffie and Hellman.

In their 1976 paper, "New Directions in Cryptography" [2], Diffie and Hellman postulated that the key management problem is solved, given public key technology, by the publication of a modified telephone directory, which they called the Public File. Instead of a name, address and phone number, the Public File would contain a name, address and public key. When you want to send me a message for my eyes only, you turn to the Public File, find my entry and use the public key associated with that entry to encrypt a message for me. Only I can decrypt that message, since presumably only I have the associated private key. Because of the nature of public key cryptography, there is no need to keep the public key secret, although one must still protect that Public File from tampering.

As a demonstration of the power of public key cryptography, this was a brilliant example. The problem is that there are people who took this example literally and set about creating such a directory, when as I point out here, there is an inherent flaw in this construction. Namely, you cannot find me in that directory. Diffie and Hellman solved the previously difficult key management problem by use of names, but did not offer any solution to the even more difficult name management problem.

In his 1978 MIT Bachelor's thesis [5], Loren Kohnfelder addressed the Public File proposed by Diffie and Hellman, noting that a networked version of this directory would have a performance problem. He proposed instead that each line item of that directory, which he identified as name (presumably login name) and public key, be digitally signed and distributed to anyone who wanted a copy, for them to hold. He coined the name **certificate** for this digitally signed directory line item. This may have avoided the problem of loss of access to the central Public File (e.g., because of network partition), but in fact it made the name management problem worse. On the other hand, no one was especially aware of that problem, so solving it was not part of Kohnfelder's requirement set.

In the 1980's, the X.500 effort set about building a directory like that envisioned by Diffie and Hellman, as a single directory to cover the world's devices and people. For authentication (e.g., to provide notation of the permission to modify an entry in the directory), that standards effort specified the X.509 certificate format, binding a public key to a **Distinguished Name (DN)**, which can be thought of as a pathname into the X.500 directory. For our purposes, it is an identifier that is intended to refer uniquely to the person who holds the key to which the X.509 certificate binds it.

Around 1990, the Privacy Enhanced Mail (PEM) effort in IETF chose to use X.509 certificates to identify mail recipients. There was a fair amount of excitement at the time over the potential of X.500 to make sense of what was already a bewildering set of people connected by the various networks (now just called "the Internet", but still quite small at that time, before AOL

experienced its user explosion). However, PEM failed because X.509 failed. Not only were there no Certificate Authorities (CAs) in place to issue X.509 certificates, the very process of choosing a DN and generating an X.509 certificate appeared to have legal connotations that at least the company where I worked at the time was not willing to accept.

To get around this failure of X.509, there was a version of PEM produced, called RIPEM that did not use X.509. It allowed the use of keys that were delivered out of band and used without certification. To provide for certification without CAs, in 1991, PGP allowed for any keyholder to sign the key of any other keyholder, under the **Web Of Trust** assumption: that multiple independent signatures on a certificate would be as trustworthy as one highly trusted signature on that same certificate, when you had exceeded some number of independent signatures, no matter how vulnerable each of those signers might be.

PGP succeeded where PEM failed, but there was still something wrong with the PKI model. In 1996, three independent efforts (SDSI, SPKI and PolicyMaker) departed from the PKI model in the same way: using a public key itself as the identifier of the keyholder, rather than some name. This has the advantage that there is no ID certificate needed to bind that key to the ID of the keyholder since the key ***is*** the ID.

3 Conventional PKI Wisdom

There has been a great deal written and discussed about PKI, but there are some frequently encountered items of conventional wisdom about PKI that this paper addresses directly:

1. that you need an ID certificate;
2. that you should get that ID certificate from a CA that protects its signing keys well (e.g., uses a vault with strong physical protection against theft or misuse of keys);
3. that with such an ID certificate, you will know with whom you are dealing when you process a signed message or encrypt a message to some key; and
4. that with all of this, you get non-repudiation, which means that the signer cannot later deny having sent a particular signed message when you present that signed message to a judge and ask for it to be considered binding against the human you have cited as the signer.

As it turns out, all four of these items of wisdom are seriously flawed, if not completely false.

3.1 ID Certificates

The original model of an ID certificate was one that would bind me to my entry in the X.500 directory, by way of the DN that both identified me and uniquely specified my entry in the directory. The assumption was that one needed only one such entry (or perhaps two: one at work and one at home).

By contrast, each of us has multiple identities both at home and at work. I, for example, have five different but equally valid IDs at work. They are used for different functions and their format and nature was determined by the applications in which they are used. At home, I have even more. There are 4 credit card numbers, 1 ATM card number, 4 bank account numbers (all from the same bank), ISP account names, etc.

There are two problems with getting one ID certificate:

1. we would have to change all legacy software and business processes to use that one ID or have that one ID certificate list all of my IDs; and
2. we would have to find one CA with the authority to establish all of those ID to key bindings.

We take it as impossible to change all business processes to use one common ID. It is also a potential privacy violation either to use a single ID or to bind all different IDs into one credential, so that some party can know how to link all of my transactions to one another.

More serious is the problem of finding one certificate issuer that has the authority to do all of these ID bindings. My company will accept only itself to bind my key to my employee ID number. My bank will accept only itself to bind my key to my bank account number. The key used could be the same in both certificates, but the binding must be performed by an entity with the authority to perform that binding. That authority comes from business rules and security policy, not from some CA characteristic like strength of protection of the CA's own private keys.

The conclusion is that we cannot have one ID certificate that is used for everything. We will most likely need as many certificates as we have relationships.

3.2 CA Key Security

It is accepted wisdom that certificates should be issued by a Certificate Authority that operates out of a vault – that is, that protects its signing keys very strongly, with military grade physical and personnel security, multi-factor authentication of people, multi-person access controls, etc. Such a facility is extremely expensive, so there cannot be many of them. Let us consider the use

of a CA in four different ways, discussed below, and improve on this design.

3.2.1 Client goes to the Vault

Early theoretical papers on certification assumed that the client would go to the vault, present credentials proving identity along with a public key and receive an ID certificate in return. This is presumably secure, but has the problem that it is too expensive for the user.

Meanwhile, it actually has a security problem, in that there will be very few such vaults, so the people running the vault have no idea who the user is. They will never have met the user and therefore will have to rely on other credentials to establish the identity of the user. This weakens the overall process to something less than the security of the credentials used and opens the process up to traditional identity theft techniques. Since we see identity theft increasing in frequency, it is doubtful that we could call this mechanism secure.

3.2.2 Client Opens a Channel to the Vault

One early attempt to overcome the expense of the previous method was to permit a client to open a communications channel to the vault. This could be by telephone, but more likely it is by web form over an encrypted channel.

Let us assume for the sake of argument that the connection is established and there is no man in the middle. We know that if you have a confidential channel, you can mutually authenticate the parties on the two ends by use of a shared secret. So, it is possible to establish identity over this channel. Once that has been done, the CA in the vault can issue a certificate for the public key provided by the user, and from then on, that key pair and certificate could be used for authentication.

The problem comes with establishing that shared secret.

At least one company considered making a business relationship with a credit bureau and then using the credit bureau's body of knowledge about the user to quiz the user and establish identity. The problem with this mechanism is that there are no secrets shared between the user and the credit bureau. That is because the credit bureau's primary business is the selling of the information it gathers about people. Making matters worse, even if one were to find a repository of information about people that is not in the business of selling that information, if it uses the same information that some other organization makes publicly available, then that information can still not be used as a secret shared with the user.

So, the problem of establishing a good, high entropy, shared secret with the user boils down to something as expensive as the first mechanism. That is, the user can come to the vault, prove identity to trusted employees of the vault, get that identity recorded along with a high entropy secret generated and shared with the user during that visit. That high entropy secret can then be used later, over a web connection, to get a certificate.

3.2.3 Registration Authorities

With the previous mechanism ruled out because it is either grossly insecure or as expensive as the first mechanism, the next step is to reduce the cost for the user by enlisting registration authorities (RAs). For each CA, there would be a large number of RAs, so that any user could find an RA within easy travel distance. The user could then prove identity to that RA. The RA would then instruct the CA, over a mutually authenticated, cryptographically secured channel, to issue the desired certificate from the vault.

This allocates the cost of the first mechanism to the CA infrastructure rather than the user. The CA has come to the user rather than the other way around. This also could have a security advantage. That is, if there are enough RAs, it could be that the user would be known personally by the RA and identity could be established not by paper or plastic credentials but rather in person. This would reduce the threat of standard identity theft.

Although this is far more secure than the previous mechanism and much cheaper for the user than the first mechanism, its security can be better.

3.2.4 CA on the RA Desk

To improve the security of the previous mechanism, the secured network connection between the RA and the CA should be severed and the computer on the Registration Agent's desk should run a CA and directly issue the user's certificate.

This is categorically more secure than the previous design, primarily because the network connection between the RA and the CA has been eliminated, depriving an attacker of one avenue for attack. There is also a security advantage, since the CA in the vault would now not sign individual certificates but rather sign the certificates of the next layer of CAs – those now on the RA desks. Because this is a much lower volume operation, the CA could operate in a different fashion. For example, it might use split-key (distributed signing) technology rather than a single, secured vault. With enough key shares, split-key technology can be arbitrarily secure, far surpassing the security of any vault, even with key shares held in only moderately secure but tamper-evident storage.

Some may argue that this design exposes a valuable key – the final CA private key – to possible theft because the RA computer is not specially protected. However, this could also be a security advantage. If an attacker

can steal the CA key from the computer on the RA desk, then that attacker could just as easily steal the key by which the RA authenticates its connection to the CA, under the previous design. Under that design, the attacker could then get the CA to sign a false certificate and that false certificate would have the imprimatur of having come from the real CA in the real vault. If the theft were discovered, then all signatures by that CA key would be called into question and the CA key itself might need to be revoked, along with all certificates it had generated. Under this last design, if a leaf CA key were stolen, then only that one key need be revoked along with only those certificates it had generated.

3.3 Know the Other Person

The third element of conventional wisdom is that with a proper ID certificate, you can know the person with whom you are transacting. This idea traces back to the 1976 Diffie-Hellman paper [2], which made the assumption that the first important job was to learn the identity of the party on the other end of a communications connection. The Public File and then the ID Certificate were to achieve that by binding the person's name to the person's public signature key.

This assumes that names work as identifiers.

3.3.1 The John Wilson Problem

The fact is that names do not work as identifiers. This has come to be known as the John Wilson problem, named after a co-worker.

3.3.1.1 E-mail

At Intel, there are (at the time of this writing) eight employees with the name John Wilson, in some spelling. The IT department is very careful to make sure that each of these John Wilsons has a unique name. That is because these names are used as e-mail addresses and to index into the corporate employee database.

In spite of the care with which each John Wilson has been given a unique name (e.g., through the use of middle initials), John keeps getting mail intended for one of the other John Wilsons and they keep getting mail intended for him.

3.3.1.2 Airport

This problem isn't limited to e-mail misdirection.

In August of 2001, John was returning from a one-day business trip to the Bay Area. He had an electronic ticket and no luggage. It was a simple trip.

On the return leg, he went to the ticket counter, was asked for an ID (his driver's license) and was asked if anyone unknown to him had given him anything to carry, etc. The ticket agent printed out his boarding pass and gave it to him. He was looking at it as he started to walk away but turned back to the ticket agent to say, "I'm not going on to Eugene. I'm just going to Portland."

The ticket agent took back his boarding pass, consulted the computer, and said that he had the boarding pass for the other John Wilson on that flight. That other John Wilson had his boarding pass.

So, the solution was for John to go to the gate and have them page John Wilson – and then, when the other John Wilson appeared, trade boarding passes.

Especially in light of the post-9/11 requirement to have luggage removed from a flight if the ticketed passenger does not take the flight, this could have been a serious security problem.

3.3.1.3 Ann Harrison

When I tell the John Wilson stories, instead of getting a reaction of disbelief or scorn at my making too much of a case out of an isolated incident, the reaction is almost always "That's nothing. Listen to this."

A friend of a friend, Ann Harrison, reacted that way.

She told of sitting on the examining table in her doctor's office, waiting for the doctor, when the nurse came in, carrying a syringe. The nurse said, "This will only sting a little". Ann asked, in shock, what the nurse was trying to inject her with and the nurse replied that it was Botox (botulism toxin). Ann said that she doesn't get Botox injections, to which the nurse replied, "but you're Ann Harrison, aren't you?"

3.3.1.4 Carl Carlson

In the early 1900's, Carl Carlson was working in a factory in Wisconsin, in a heavily Swedish community, and was getting annoyed that he kept getting paychecks for another Carl Carlson, one who earned less than he did. So, sitting in the bar after work one payday, he decided to change his name to something really unusual and avoid this problem. He looked across the bar and saw a sign with a really unusual name … and that's how my ex-in-laws ended up with the family name Miller.

3.3.2 Names are not IDs

These anecdotes illustrate a point that should be of concern to us as computer scientists and especially to those of us involved in PKI.

Human beings do not use names the way we want them to.

The actual process by which humans use names and the psychology behind that process deserve a great deal of study. It is clear, even prior to that study, that computer

developers and computer users deal differently with names.

I speculate that computer developers, and especially PKI or large directory developers, think of names the way we do variable names or file path names. That is, a name is some string, unique within its block or directory or context, that unambiguously identifies some object (value, file, person, …) – and we further assume that the mechanism that uses this name (a compiler, an operating system, or a human user) will follow that unique string to the same object any other mechanism would follow the string to.

Compilers and operating systems may behave this way, but **human users do not**.

Our PKIs assume they do. Our mail agents assume they do. Much of what we design in computer science makes this same, false assumption. For our immediate concern, the main impact is that PKIs are based on a false assumption and the security of systems using those PKIs suffers as a consequence.

In a way, however, this is good news. This means that there are a great many fresh new research opportunities. For example, how would you build a mail agent that does not use names or e-mail addresses for people?

3.3.3 ID as Dossier

It is doubtful that human beings could ever be trained to read all information offered in a certificate and verify it against their knowledge of a person, before jumping to a conclusion about the identified person. Even if that training could be achieved, however, an ID certificate usable by everyone would become a dossier.

Consider an ID cert for John Smith. The name alone doesn't tell you which John Smith, so you need additional information. Andy works with John, so he needs John's employer (and building and mail stop) in the ID certificate. Betty knows John only at home, so she needs his home address in the ID certificate. Charles knew John at work 10 years ago, so he needs John's work address from 10 years ago. Dan shared a hospital room with John back in 1994, so he needs a record of John's hospitalization from then in order to identify John unambiguously. This process needs to be iterated over all possible relying parties, to make sure the ID certificate works for all of them.

The result would be a nearly complete dossier on the keyholder, and that dossier would almost certainly violate privacy laws, not to mention John's desires. As a result, the ID certificate could not be released to the public. That, however, violates the basic purpose of the ID certificate. A workable alternative would be to have different ID certificates for use by different relying parties [6], but that violates the design goal of one ID certificate that lets an arbitrary relying party know with whom she is transacting.

3.4 Non-repudiation

The fourth item of common wisdom has to do with non-repudiation, which is usually defined as the inability of a person later to deny having digitally signed a document.

The central idea behind the concept of non-repudiation is deferred enforcement of security. That is, one receives a digitally signed document (often described as a contract, when non-repudiation is discussed) and verifies the signature on the document and the certificate chain that identifies the key used, and then acts on the document. In most cases there will be no intention of fraud and the transaction proceeds normally. However, in case there was fraud, the document can be produced along with its certificate chain to present to a judge. The judge can verify those signatures and thus establish that this document was signed by the defendant.

There are several problems with this understanding and this process.

3.4.1 Expense

The process described above is expensive. The digital signature and certificates that bind the signer to a document do not bind that signer to a location. The signer must be located and brought to trial. The process of location and the process of trial are both expensive. If the amount of the loss were small enough, taking the case to trial would not pay.

3.4.2 Not Adequate

Assuming non-repudiation was achievable, technically, and a judge found a defendant responsible, this process works only if the victim can be made whole. In cases of moderate financial loss, this might be adequate. However, if the loss were of something more valuable than the perpetrator's total lifetime worth, then the victim cannot be made whole. Worse, if the loss were of a life or of secrets, then no amount of money could compensate the victim.

3.4.3 Not Achievable

The main problem with the theory of non-repudiation is that it is not technically achievable. That is, the intention is to bind a human being to a digitally signed document. With a holographic signature on a paper document, the human's hand came in contact with the paper of the document. With a digital signature there is machinery between the human and the signed document: at least a keyboard, software (to display the

document and to drive the signature process) and a key storage and use facility (e.g., a smart-card).

No one has demonstrated, in the normal computer for home or office use, the prevention of introduction of hostile software. To the contrary, we have seen a steady increase in such incursions over the years.

There are secure facilities for key storage and use, but no mechanism that an average home or small business user would choose to buy has been proved secure.

Meanwhile, computers are not restricted to isolated rooms with card access entry, raised floors, guards outside the glass walls, etc., that they might have been in the 1970's when much of this thinking about public key cryptography had its nascence. Computers are not only everywhere; they are unprotected to a continually increasing degree. Therefore, even if the computer has no hostile software and its private key is kept in a truly secure facility, access to the keyboard of that computer is not limited to the person certified to be associated with that private key.

What might make this process of non-repudiation work would be hardware that would serve as a witness to a signature, providing tamper-proof evidence of the actions of a human being (e.g., through videotape), of what that human was reading and of the human's positive action to assent to the displayed document. Such a log of human behavior could then be presented in court to prove the claim of non-repudiation.

Of course, if such hardware were available, then we would not need digital signatures, much less the assumption of non-repudiation on digital signatures.

3.4.4 Contractual Commitment

For lack of technical achievability, some people try to legislate non-repudiation. If laws are written to presume that the certified keyholder is responsible for anything done by that key, then the rational thing for a computer owner to do is to refuse to accept ownership and use of that private key. That could bring not just PKI but use of public keys to a screeching halt.

The good news in this is that we do not need non-repudiation in order to do business with digital signatures. If two parties want to do electronic business with each other, they can sign a paper contract with one another in which party A might declare that it would honor any document digitally signed and verified with a public key that is given in the contract (or whose cryptographic hash is printed in the contract). The party accepting that responsibility for that key could then protect that key with mechanisms appropriate to the way that key was empowered. If one is ordering office supplies with that key, then maybe it is kept encrypted by password on the hard drive of a PC on a secretary's desk. If one is ordering millions of dollars worth of industrial supplies, then the key might be kept in a locked room, under 24x7 guard, with multi-factor authentication for people entering the room, special computers with strong key storage facilities that erase their keys if the mechanism is physically moved, no network connections for the computers and strict control over the software that is allowed to be loaded onto the computers.

4 New PKI Wisdom

The reasoning above gives us a new list of PKI Wisdom:

1. There is and will be no single ID, so a single ID certificate makes no sense.
2. Discard RAs and put CAs on the RA desks.
3. Knowing a keyholder's certified name does not tell you who that keyholder is.
4. Non-repudiation is neither adequate for serious problems nor achievable.

So, instead, we need to do strong access control and that requires more than ID certification. There are several ways to achieve access control, as outlined below.

5 Certificate :: DB Trade-off

As we consider the various ways to do access control, we must address the religious battle between those who advocate certificates and those who advocate servers. Each technology can achieve the same results, under certain assumptions. The main difference is in their behavior under network load or partition, but there are security differences, discussed later in this paper, having to do with database administration.

For example, Kohnfelder created certificates by digitally signing a line item from a protected database: the Public File. This has the advantage of making verifiable data available even when the database is not, whether by network partition or by mere performance problem.

This process can be applied with any kind of database. In particular, it applies to all three edges of the credential triangle shown in Figure 1.

5.1 CAP Principle

Fox and Brewer of UC Berkeley have put forth the CAP Principle [4], stating that it is possible to design a distributed system that achieves any two of:

1. **C**onsistency
2. **A**vailability
3. tolerance of network **P**artitions

but it is not possible to achieve all three.

The invention of certificates as signed line items from the Public File was a choice to achieve A&P while the Public File achieves C&A.

There are frequent attempts to criticize one or the other of these mechanisms for not achieving the third desirable attribute and to come up with some new design that tries to achieve all three, but by the CAP Principle such attempts are doomed.

One must look at the specific security requirements of a particular application and decide which of the three desirable attributes can be sacrificed. This choice will be different for different applications.

6 Credential Classes

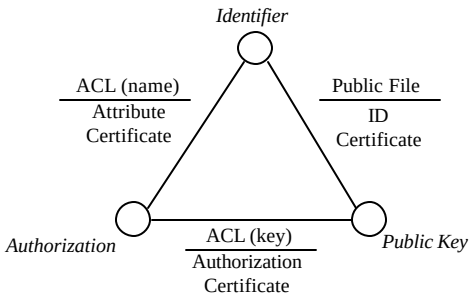

Figure 1: Credential Classes

Diffie and Hellman bound *Identifiers* to *Public Keys* through the Public File. Kohnfelder took line items of that public file and made ID certificates.

Those of us who wanted to use ID certificates as part of implementing access control, needed to get from *Authorization* to *Public Key*. That is, a transaction would come over the net with a digital signature verifiable by a public key and it would require authorization before it could be honored.

The knee-jerk reaction, relying on time-sharing system practice from the 1960's, was to use an Access Control List (ACL) binding authorization to login name. [By the way, Kohnfelder described the names in his thesis as login names, so this use of an ACL is not mixing metaphors.]

By the arguments of section 5, you can also convert line items of the ACL into certificates, and in this case, they become what we know as **attribute certificates**.

In 1996, however, a number of us started developing the third side of the triangle: **authorization certificates**. That is, something directly binding an *authorization* to a *public key*, rather than going through an *identifier*.

Also, by the logic of section 5, one can have protected database versions of the authorization certificate as we find with X9.59 and with the SSH access control file (.ssh/authorized_keys).

7 Authorization via ACL and ID

Figure 2 shows the use of an ACL and ID certificate to determine authorization. The ACL could be held locally in the machine that acts as gatekeeper for the protected resource, or it could live in some central authorization database that the gatekeeper queries over the network to approve any access request.

The security perimeter shown in Figure 2 indicates that both elements of the process – the ACL (or attribute certificate) and the ID must be protected equally. If the attacker can control either, then he or she can get improper access. However, there is a third vulnerability not immediately visible in the triangle diagram: the name. That is, the diagram shows one "Identifier" node at the top of the triangle, but in fact there are two identifiers involved: one on the ACL edge and one on the ID edge. The identifiers need to be the same, to link these two sides together, and some mechanism has to do the comparison to establish that.

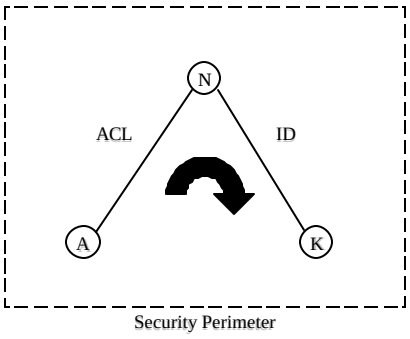

Figure 2: Authorization via ACL and ID

If that mechanism is executed by a computer and the names used are unique, then the comparison can be done with security. If the mechanism is executed by a human, then even if all names are unique, the John Wilson problem shows us that there will be mistakes made, and a clever attacker can exploit those mistakes to gain improper access. A human might make that comparison with each access, as we see with S/MIME or SSL, since in those cases the ACL is kept in the human user's own head. Or the human might make a name comparison when some database is administered by a human or a certificate is issued. In general, it is safe to assume a human will be involved at some point in the process because it is for human use that names are used in the first place.

When the method of Figure 2 is used, there is also the problem of administering the ACL side of the triangle. We consider two possibilities for that, below.

7.1 Authorize Everybody

The job of building an ID PKI is difficult enough that some people rebel against building an ACL as well. Instead, they use a one-line ACL: (*). That is, grant access to anyone who has an ID certificate. This isn't exactly the non-repudiation case, since it's not a question of having a signed contract. Rather, this is a situation like that employed by browsers when they decide whether to show the padlock icon as locked or unlocked. The icon is shown locked if the ID certificate is valid (and refers to the domain name from which the web page (or part of it) came).

The problem there is that users rely on that closed padlock rather than on a personal inspection of the ID certificate to decide whether to trust the web page and its server. This leads to a wonderful quote, from Matt Blaze, in the hallways of the RSA 2000 Convention: "A commercial PKI protects you from anyone whose money it refuses to take."

7.2 Authorization DB

You can, instead, build a real authorization database. Consider the database for something the size of a large PKI, with 6 million users.

If each user changes his or her entry in the database every two years, then there is one change to the database every 2.5 seconds of each normal workday.

Since this database is being kept in a central, secured location, it is being maintained by a staff of people cleared to enter that facility. Those people do not know all 6 million users. So, when a request comes in to change the authorization of some user, it must be investigated. If that investigation were to take a man-week, then the office would need more than 50,000 investigators, making this a very large operation.

No matter how large it is, the process begs the question of what makes these people administering the central database authorities on the data they are entering.

8 Direct Authorization

Another option is to go the other direction around the credential triangle, as shown in Figure 3.

In this process, there is only one point of attack, rather than the three of Figure 2. One would have to attack the authorization certificate issuer (or the maintainer of the authorization-to-key ACL).

One might ask why Figure 3 shows an ID when that ID is not used as part of the authorization process. The reason it is there is for forensics.

One can easily gather an audit log with entries identified by keys used (or their hashes, as more compact identifiers that are still globally unique). From processing those audit logs (or other tests) one might determine that a given keyholder (a given key) has misbehaved and needs to be punished. As Steve Kent quipped, during a DIMACS talk on this topic, 'You can't punish a key. What would you propose doing? Lop a bit off?'

You need to punish the keyholder. The simplest punishment is to put that key on a local black list. That keeps the keyholder from gaining access at the machine where you discovered the misbehavior. However, you might want to actually punish the keyholder, legally. For that, you need to locate the keyholder. So, you need a link from the key to the keyholder. This is indicated as an ID or name, but more likely it would be a whole file of information that would allow a security officer, lawyer or policeman to find the keyholder. This information could include the keyholder's name, address, phone numbers, bank accounts, friends, family, employer, etc.

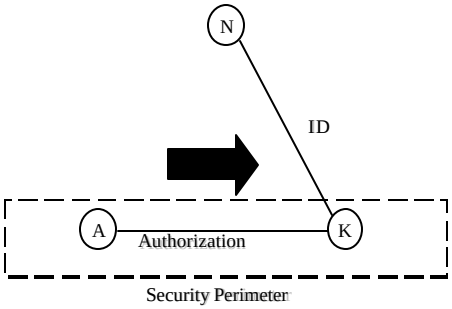

Figure 3: Direct Authorization

More interesting for those interested in PKI is the fact that this information binding a key to ID does not need to be either online or in certificate form. It is not used in the authorization process. It is used only during the manual process of punishing the errant keyholder. Therefore, the information could be kept in a non-networked PC in the security office. It could even be kept in manila folders. This affords the user with a certain amount of privacy. The user's identifying information need not be released to a resource guard whenever an access is made.

9 Delegation of Authorization

SPKI [7] permits delegation of authorization. SDSI [6] permits delegation of group membership. For some cases, the two mechanisms can be shown to be equivalent. The examples below can be achieved either way, but they will be described as authorization

certificate delegation – and contrasted with the use of a corporate authorization DB together with PKI for ID, according to the model of Figure 2.

10 Large Company VPN Example

In this example, we deal with a large company that permits VPN access only to authorized employees. We consider it two different ways, first via a central authorization database and then by distributed, delegated authorization.

10.1 VPN Access via Central DB

Figure 4 shows part of an organization chart for a large company that has decided to give VPN access only to approved employees. We assume that employees are identified by some ID PKI, but authorization is maintained by a corporate authorization database. That database is maintained by some person or group, labeled A in the figure. A user, U, requests access by web page, since A and U are probably in different states if not countries and have never met one another and are not likely ever to meet one another.

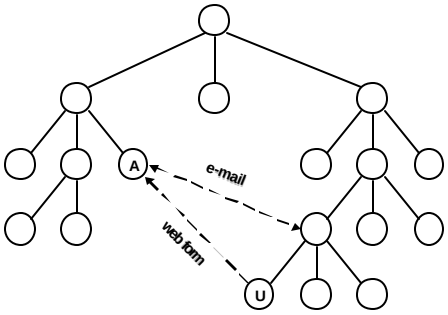

Figure 4: Central Authorization DB for VPN Access

If A were simply to enter U in the database in response to the web form, then there is no security to speak of in the system. So, A looks in the corporate central employee database to find U's manager and sends an e-mail, asking if U should be allowed VPN access. When the answer comes back in the affirmative, A enters U in the authorization database and U has VPN access.

There are at least two problems with this mechanism:

1. A sends an e-mail to someone whose name is very much like the name listed in the employee database as being U's manager. Thanks to the John Wilson problem, that does not mean that A sends an e-mail to U's manager.
2. The mechanism as described above implicitly grants every manager in the company the power to grant VPN access. Correction of that limitation would greatly complicate the database administration process.

In the next section, we address these problems.

10.2 VPN Access via Delegated Direct Authorization

In Figure 5, we accomplish the same function, but by authorization certificate and delegation of authorization. The organization or person, A, responsible for the ACL of the machine(s) that enforce VPN access, enters a public key into that ACL, as the head of a tree of certificates to be empowered to have VPN access. Person A then uses the matching private key to grant authorization certificates to his or her manager. That authorization flows, by authorization certificate, up the organization chart to the CEO and from there down the entire organization, but only into those groups where VPN access makes sense. In particular, as shown by the heavy lines, it flows from A to U and therefore has the same effect as the process shown in Figure 4.

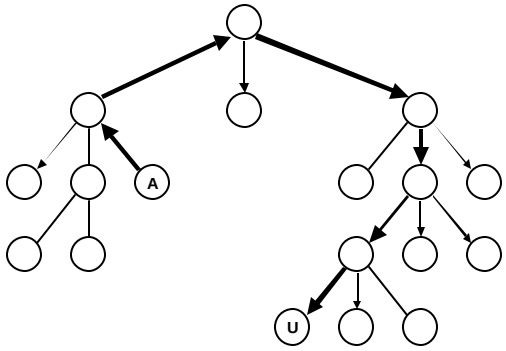

Figure 5: VPN Access by Direct Authorization

The process of Figure 5 has some distinct advantages over that of Figure 4:

1. Each grant of authorization is between two people who work together and therefore can authenticate one another biometrically, in person. Names are not used in the process, so there is no security flaw from the John Wilson problem.
2. Each grantor of authorization is in a position to know better than anyone else whether the grantee should receive that grant of authorization.
3. These decisions – of authentication and authorization – are made with almost no effort. No investigation is required.
4. The work that used to be done by A is now distributed around the company, although it is miniscule at each place a decision is made. This frees A to do other, more interesting

work. That, in turn, saves money for the corporation.

So, this process both saves money and increases security of the administration of the authorization process.

11 Cross-company B2B P.O. Example

The example of the previous section dealt with operations within a single company that had a single PKI. We now address a pair of companies that want to do electronic purchase orders, with orders automatically processed by computers in company A when they are signed by authorized keys (keyholders) within company B. Each company has its own, independent PKI.

11.1 B2B via Central DB

In Figure 6, we build a structure analogous to Figure 4. The employees of Company B that should be authorized to sign electronic purchase orders are shown in gray, while there is one person (or group) in Company A that maintains the ACL on the machines Company A uses to process purchase orders automatically.

The purchasing agents must request, somehow, to be added to the ACL, and the maintainer of the ACL needs to verify the propriety of each such request. This request goes from company B to company A. The verification of that request is a dialog initiated by the responsible parties in company A.

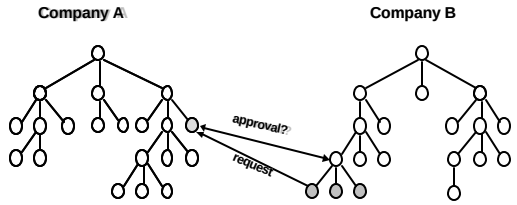

Figure 6: B2B via PKI and Authorization DB

11.1.1 Bridging of PKIs

The first thing we observe is that for ID's issued by Company B's PKI to be usable within Company A, we need to bridge the two PKIs, either with a bridge CA or by adding each PKI root to ACLs in the applications on both sides. However, when we bridge the two PKIs, we make the John Wilson problem worse for both.

1. It is made worse just by having more people under the same namespace. This leads to more name collisions and more mistakes.
2. It is possible that name uniqueness is violated. Company A could have been very careful to have only one "John Q. Wilson" and Company B could have been very careful to have only one "John Q. Wilson", but after the bridge, there are two. What is missing is some entity that would control the issuing of names within companies A and B, before they decide to bridge their PKIs. There is no such entity today, and the experience of ICANN (The Internet Corporation for Assigned Names and Numbers and other Top Level Domain efforts) suggests that no such entity will ever exist.

11.1.2 Employee Data

In the process of Figure 4, the maintainer of the ACL consulted the central employee database to find the party to contact to get approval of the request for authorization. Company A does not need the entire employee database of Company B, but it does need enough of that database (or remote access to a view of that subset) to permit it to make the proper authorization decisions.

This kind of data, especially linked to names, is traditionally considered confidential by companies. A special exemption would have to be made in this case. Meanwhile, the data that company A needs would have to be made available under strict access controls, and the authorization database for those access controls becomes an additional problem to address. This way leads to uncontrolled recursion.

11.2 B2B via Delegated Authorization

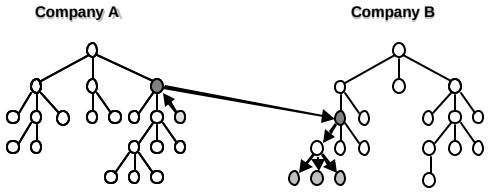

Figure 7: B2B by Delegated Authorization

In Figure 7, we show the same B2B process, but by delegated authorization rather than authorization database and ID PKI.

In this figure, we introduce a new node color (darker gray) to stand for the executives of the two companies who meet to decide to form the business relationship. These executives exist already and perform this function. Two companies do not spontaneously decide to do business with each other. There is a period of investigation and decision-making before that decision is made. The decision is usually sealed with a contract and the contract is signed by individuals of the two companies. These meetings might be electronically intermediated, but they are meetings of people rather than of computers.

In Figure 7, the permission to delegate the authorization to have purchase orders accepted and processed automatically is granted from the person or group that maintains the gate keeping machines in Company A to the executive in Company A who is going to sign that B2B contract. After the signing of that contract, the executive from A grants the executive from B the power to authorize such purchase orders. The executive from B takes that authorization back to Company B and delegates it to the purchasing group manager who certifies the individual purchasing agents within her group.

Note that this process:

1. does not use a bridge CA, so it saves that expense,
2. does not use names, so there is no John Wilson problem,
3. does not require either company to access the other company's confidential employee data,
4. does offer improved security, just as we saw in Figure 5.

12 The AND Effect of ID PKI

There are those who claim that doing authorization computation via the combination of ACL and ID cert is important because it gives you a logical AND of two conditions: the authorization and key validity. The assumption there is that a valid ID cert does more than name the keyholder. It also represents certain security conditions. It attests to the key itself not having been revoked and might also attest to the keyholder's continued employment.

This is valuable functionality. However, the use of an ID instrument for these other characteristics is not the best system design. What if some application cares about key compromise but not about continued employment? This mechanism does not allow the application designer to separate those three attributes of a key: ID, non-revoked status and continued employment. It also does not allow the application designer to specify the AND of other functions, without loading those onto the ID instrument as well.

A cleaner design is to use an explicit logical-AND and specify the conditions individually, each with its own certificate (chain). Each of these attributes can be bound to a key by an authorization certificate, with the certificate issued by the proper authority. That is, a 24x7 key loss reporting service might be in charge of providing online validity information of the non-revoked status of a key while a corporate HR office might provide information about continued employment. These attributes do not require any ID. They can be bound directly to a key. By contrast, loading all of these attributes into an ID certificate by side effect requires the ID certificate issuer to be the authority on all of those attributes – something that may be difficult to achieve, organizationally.

[Note that SPKI/SDSI [7] includes a construct called the "threshold subject" that permits expression of such "AND" conditions in ACL entries or certificates. The code that implements threshold subjects is available in [1].]

13 Conclusions

This paper makes the case that there are fundamental problems with the original ID-based notion of a PKI, in that it fails to take account of certain realities (such as human limitations). Instead, we can use delegated, distributed authorization, which does not suffer from those fundamental problems. Two examples of the use of distributed authorization were given, in brief, but there are a great many other examples. The reader is encouraged to try applying these techniques to other problems, as was done in [3].

14 References

[1] CDSA: http://developer.intel.com/ial/security - source code and documentation, including a full implementation of SPKI and SDSI certificate reduction. This link leads to the open source repository for that code.

[2] Whitfield Diffie and Martin E. Hellman, "New Directions in Cryptography", IEEE Transactions on Information Theory, Vol. IT-22, No. 6, November 1976.

[3] Steve Dohrmann and Carl Ellison, "Public Key Support for Collaborative Groups", Internet2 PKI Workshop, April 2002.

[4] Armando Fox and Eric A. Brewer, "Harvest, Yield, and Scalable Tolerant Systems", Proceedings HotOS-VII, 1999

[5] Kohnfelder, Loren M., "Towards a Practical Public - key Cryptosystem", MIT S.B. Thesis, May 1978.

[6] SDSI: http://theory.lcs.mit.edu/~cis/sdsi.html

[7] SPKI: http://TheWorld.com/~cme/html/spki.html