

# Improving Message Security With a Self-Assembling PKI

Jon Callas (*PGP Corporation, Palo Alto CA, jon@pgp.com*)

April 4, 2003

## Abstract

Public Key Infrastructures (PKIs) exist for a number of purposes. One purpose for a PKI is achieving wide-spread deployment of secure communication; the PKI makes it easy for two parties to communicate securely. Another purpose is that of secure delivery to a recipient; the PKI makes sure that when Alice sends a message to Bob, it arrives to the very Bob that Alice intended it to arrive to. The paper describes a *Self-Assembling-PKI*, a new way of constructing the corpus of certificates that makes up a PKI with a solution to the first problem – widespread deployment of secure communication. It does not address the second one, but can use and interoperate with a PKI designed to achieve that goal. It creates itself by observing and monitoring existing message traffic, and transparently inserting security protocols into existing traffic. Most radically, its mechanisms trade perfect security for ubiquity.

## 1 Introduction

The *Self-Assembling PKI* is not a new technology, nor does it require new standards; instead it is a new way to think about existing PKIs, security standards, and systems to achieve these goals:

1. Wide-spread deployment of communications security. Presently, within organizations, secure email tends to be used by five to fifteen percent of the organization [PGPUSE]. This figure applies to a self-selecting group that includes people who already consider secure email to be important or have regulatory requirements. Network-wide, this number is much lower, probably no more than five percent of total users [GARTNER]. A number of factors cause this, all of which ultimately center around ease-of-use [JOHNNY].
2. Transparency of use. Even within a population of people who regularly use traditional public-key cryptography, human error is a formidable problem. The solution to these inevitable lapses in judgment is a system that requires no thought on the part of the user.
3. Ease of deployment. Not only must a PKI be easy for the end users to work with, but it must be easy for its administrators to set up, deploy, and run. Traditional PKIs have not been widely deployed, difficulty in deployment being much of the problem [GUTMANN]. This is another facet of ease of use, merely with another set of users.

4. Policy management. A PKI that is operating on behalf of end users must have a collection of rules that describe what actions it performs under what circumstances. These rules are the PKI policies; they describe both the mechanisms that apply to outgoing messages and incoming messages as well.
5. Risk Mitigation. Because the PKI is operating on behalf of humans and without their intervention, it is vital that it include detailed reporting. Inevitably, the PKI will contain errors, and its human administrators must themselves detect and correct these problems. This cannot happen without robust reporting.

Additionally, there is an old principle of security that the value of what is being protected affects the measures taken to protect it. This approach strives to the above goals, but is willing to trade perfection for ease of widespread deployment in the belief that a message security system with known limitations that can be used by anyone is more secure overall than one that can only be used by a few experts.

## 2 A Change in Metaphor

Traditionally, a PKI operates using a telephone metaphor. From the very first description of certificates, they were described to be analogous to a telephone number; Alice would find Bob's certificate similarly to the way that she might find his telephone number.

The Self-Assembling PKI is the metaphor of the *robot operator*. The PKI determines the connection and relationship between the sender and recipient, and processes it. Without prior registration into the PKI, however, there is no way to complete the connection – if you don't have a phone, you don't have a number, and no one can place a call to you.

The Self-Assembling PKI uses a different metaphor. It is a postal metaphor rather than a telephone metaphor. This operates as if it were a *robot messenger* that has the job of delivering information as securely as possible. The messenger operates on behalf of the sender and recipient, adding security to the system (which would otherwise be done with standard insecure protocols) as much as it can. It uses the policies and heuristics of both the sender and recipient to improve a transaction between them.

We look at the problem this way because of the realities of how users use Internet communications. When someone sends an email message, they expect it to be delivered post haste. Security is a feature they desire, but delivery is what they are after. Similarly, people rarely stop using other systems such as web browsing and instant messaging just because they are reasonably insecure. In fact, one of the main security concerns organizations have about instant messaging in particular is that users shift to it because it is fast, reliable, and convenient. Instant messaging sees great growth in organizations where email security policies make email inconvenient.

People need to communicate more than they need to communicate securely. As security system designers, we may not like this, but it behooves us to be their messengers rather than their switchboard operators. If we refuse to connect their call, they don't decide they didn't need to place the

call, they just find another way to do it. The messenger metaphor is the attitude that the mail must go through. It is a step towards making PKIs be used when they haven't in the past. It makes PKI be an enabling rather than disabling technology.

While the messenger metaphor applies most closely to email, it also fits other protocols and communication systems. I have already mentioned instant messaging, but the principles apply to many other systems as well.

### **3 How the PKI Self-Assembles**

The whole point of a self-assembling PKI is of course that it does not require its administrators to construct it before it can be used. Often a PKI requires the people who construct it to understand the larger system it resides within. If they must do this while they are building it, it dramatically slows down construction.

Compounding this difficulty, even though the resulting PKI might in theory be completely accurate, the externals may have changed. If it takes longer for the PKI to be constructed than for the system it serves to change, then the PKI will never be accurate.

Self-assembly shifts the PKI staff from constructing the infrastructure to overseeing it. They correct inaccuracies, shape policy, and adapt the PKI to the larger system it serves. It is similar to the manufacturing principle of continuous improvement. It combines the great power of computers to rapidly, accurately do repetitive tasks with the power of humans to understand complexity and provide feedback to mechanism.

Components of PKI construction work by getting in the middle of the network processes, monitoring them, observing them, and constructing the PKI so that it reflects the actual use of the system.

Other components of PKI work within the active network processes, shaping them and adding security features. For example, email can be encoded to have a security envelope. An instant message can also be wrapped with added authentication and message privacy.

It is also important to note that a pre-existing PKI only enhances these newer, more flexible components. This need not replace existing PKIs. The robot messenger can exploit the work done to create robot operators.

#### **3.1 Format Agnosticism**

The robot messenger desires to deliver messages, and desires to deliver them as securely as possible, despite obstacles. A physical messenger must overcome rain, snow, and dark of night. The robot messenger has to overcome a wide variety of security standards including OpenPGP, X.509, S/MIME, SPKI, XKMS, TLS, and so on.

The robot messenger must therefore be without religion when it comes to message and certificate format. It must be able to speak a variety of protocols well enough to be understood, and well enough to abide by the policies that govern their use. The mail must go through.

While there are quite a number of possible combinations, navigating them isn't as difficult as it could be. If the messenger finds an OpenPGP certificate in a global keyserver, the recipient probably would prefer the message to be delivered in OpenPGP format rather than S/MIME. Similarly, an X.509 certificate in an LDAP directory probably calls for an S/MIME message. It's a safe bet. Heuristics do work.

Ironically, some incompatibilities can allow for better heuristics as well. There are a number of issues around using LDAP as a mechanism for distributing certificates [CHADWICK], but while the lack of a unified directory mechanism makes lookup harder, it also provides hints as to how to use the certificate.

Nonetheless, even when the system can completely infer how to use a certificate, implementing format agnosticism has a few rough edges that the implementations must overcome. Here are two obvious ones:

### **3.1.1 Multiple Certificates**

The messenger might find multiple certificates for a given recipient. It is also possible that at least one of these certificates might be bogus, expired, or lost. Policy and heuristics can assign value to the certificates by weighting the authority of a CA, timestamps on the certificates, size of the key, and so on. While the general case can have many options, these can be truncated with obvious shortcuts such as using a certificate if supplied by some reasonable authority such as the recipient's domain.

In other cases, such as finding multiple OpenPGP certificates, a heuristic could be to use them all, or a reasonable subset. Note that in this case, there is a security issue. The issue is that Alice's message may be encoded to Bob but also to an eavesdropper, Eve, who is impersonating Bob. So long as the message is kept out of Eve's hands, the security of the message is preserved. In the general case, this can be guarded against with relatively simple mechanisms within the infrastructure – such as protecting the actual transport of the message via SSL/TLS, IPsec, or SSH. There are still cases where this does not guard against Eve, for example the case where Eve is the sysadmin of Bob's mail server. The infrastructure can help protect Bob in future messages, and some of these are described below.

The most counterintuitive situation is when the messenger finds two incompatible certificates of equal perceived value (for example, a X.509-S/MIME certificate and an OpenPGP certificate). Following the principle that the mail is to be delivered, the messenger could send the same content in two separate messages.

### **3.1.2 Multiple Recipients**

Messages are often sent to a group of people. This creates similar issues to the section above, but with a small added twist. For example, if Alice is sending a message to Bob and Cindy, it may simply need to be coded in S/MIME to Bob, but XML-encrypted to Cindy.

## **3.2 Certificate Creation**

The Self-Assembling PKI can use existing certificates, but to achieve its goals, it must create keys and certificates for all of its users. As mentioned above, it sits within the network infrastructure. A number of components of the PKI proxy existing protocols as part of their work. In this position, they can observe the appearance of authenticated users, and automatically create certificates. These certificates can be rewritten as more information is learned about the users.

The PKI can manage the certificates it creates for the users, or it can share them with the users for joint management. (There is an obvious third case in which a user creates their own certificate or has a key certified by a CA. For these purposes, this case is the same as using an existing PKI's certificates; this can be considered to be merely be a PKI of a single user.) PKI-managed certificates may be marked as being in the possession of a machine. Depending on policy, they may be considered lower-valued certificates than ones held solely by the end user. This is perhaps more important for a certificate used to sign rather than encrypt, but many people are uncomfortable with the notion of robot-controlled key pairs, and so we allow for (and encourage) full disclosure.

A few examples of how the PKI creates and manages certificates follow:

- Alice connects to her usual mail server over the POP3 protocol. A proxy mediates this connection, and upon observing her successfully authenticate to the actual mail server, creates a certificate for her.
- Alice sends Bob a mail message, which is itself authenticated using SMTP-AUTH. Part of the message, the "From" line of the message has Alice's full name. Her certificate gets updated to contain that common name. Alternatively, an LDAP company directory might supply personal information for that certificate. Since Bob is a user on the same mail server, the PKI creates a certificate for him. It encrypts Alice's message using the key in that certificate and sends it on to the mail server.
- Bob connects to his usual mail server over IMAP4. The same proxy mediates this connection, and when Bob reads his message from Alice, the proxy automatically decrypts it. Policy can govern whether this is wholly transparent, or whether the message is further modified to let Bob know that it was delivered securely.
- In further work with the server, rather than the clientless operation described above, software on either Alice's or Bob's computer could share the key with the server and decrypt the message locally.

- Alice could inform the PKI server (through a web browser or other means) that she prefers managing her secure messages herself. She gives the server a certificate with which it may encrypt her mail. Note that it is also possible for the server to infer this itself.

There are also opportunities for a portion of the larger PKI to use policies and strategies beyond the usual. Here are some examples:

- A team of support specialists in an operations center share the same key within their certificates. This key is changed every month, but all specialists have the same public key so that a workflow system can route tickets to any given support person.
- A high-security engineering team uses a variation of OpenPGP enhanced to support Perfect Forward Secrecy [PFS]. Their public encryption keys are essentially ephemeral, and part of the back end mail system manages the message security with paired FIPS 140 level 4 hardware security modules, and three-factor authentication on the engineers' laptops.

### **3.3 Side-Stepping Revocation**

Certificate revocation is the hardest, stickiest, least well-implemented, and arguably most important part of managing certificates. There are numerous systems where revocation has simply been ignored and unimplemented.

Just as a Self-Assembling PKI can use an existing PKI, it can use an existing revocation scheme with CRLs, on-line checks, etc. However, within its own domain, there are pitfalls to avoid, and benefits to be gained by rethinking the part of certificate life-span where the certificate ceases to be valid.

There are also known ways to ameliorate, if not eliminate the revocation problem. SPKI [SPKI] uses the clever mechanism of simply declaring that certificates cannot be revoked. OpenPGP [OPENPGP] has revocation information travel as part of the certificate itself, with obvious advantages and disadvantages. The most important disadvantage is that it is possible for out-of-date or hostilely modified certificates to be missing revocation information. Revocation lists provide an authoritative place to get revocation information, but are vexing in many dimensions. Much work has been done on ways to eliminate them.

Being an on-line system, but one that should operate without human intervention, the SPKI solution of waving away the problem has many advantages. However, the pro, combined with some principles [RIVEST] that value new certificates over older ones, along with pushing the responsibility for certificate validity in two directions. First, the party accepting the certificate has the responsibility to decide if a certificate is good enough, and the party issuing the certificate has the responsibility to construct a certificate that the acceptor likes.

In the general case, there are potential problems with certificates that are close to expiring, as well as ones that have excessive life. However, these concerns are ones that any given messenger

may solve with its own rules. A messenger delivering high-value transactions will be pickier about certificates than one delivering chit-chat.

For our purposes, a productive and easy-to-deploy framework of the Self-Assembling PKI uses short-lived certificates. This tends towards the SPKI model, even when the data format of the certificate is X.509 or OpenPGP. It uses the validity timestamps in these certificates as freshness markers [STUBBLEBINE]. This way, internally generated certificates presented to the outside world will have a limited life, and if the keys in the certificates must be truly revoked, any “suicide notes” also have a limited life. By policy, the nominal life of the certificates managed by PKI servers range from weeks to minutes.

Additionally, these certificates may be in more detailed states than simply being valid or invalid. They could be inactive – not presently valid, but able to be re-enabled at any time. Certificates can be permitted to expire if they are not used, reported on if not used, and this can be part of the oversight into other infrastructure pieces. For example, if an employee leaves a company and the IT staff aren’t told about this, inactivity of the user’s certificates can alert the staff to this fact. Similarly, if a PKI discovers that a user’s account on some server is no longer active, it can immediately remanufacture that user’s certificate as expired, and alert the PKI administrators of this inconsistency in the infrastructure.

Certificates could also be *revocably revoked* – in effect, if not in actual syntax. It is not uncommon, for example, for an organization to regularly use the same contract staff for short stints. It is also unusual, but not unheard of for an organization to sack an employee in downsizing, only to hire that person back as a short-term contractor. As much as this situation might make security architects flinch, the *business reality* is that some important decisions don’t simply happen. I might hand someone back their badge for three months. If I do, I need to hand them back their certificates as part of that “badge.” The messenger metaphor is the idea that the system must work as well as possible. It is the idea that the mail must go through, the show must go on.

Reality thus gives us many reasons for remanufacturing certificates with short lives. Not only does it allow us to finesse many problems of revocation, these short-lived certificates in a flexible PKI make it possible to create adaptive solutions to real-world problems.

### **3.4 Certificate Trust and Search**

We discussed some of the features of certificate use and search above when we discussed format agnosticism. However, much of the new thinking in Self-Assembling PKI takes place in the use of certificates.

Let us invoke again the messenger metaphor. The PKI, acting as a messenger for the user, makes a delivery to someplace else. Let us assume that the recipient has a suitable certificate, we merely have to find it. Secondly, we need to differentiate between members of a set of certificates with varying validity. Which one(s) of those will we use? This is of course, again, a matter of policy, but policies need to be simple to create and understand. Fortunately, there are a number of simple ways to get flexible and simple search, validity, and trust systems.

Trust policy and search policy are closely related, but not the same. Search policy states where to look and in what order, trust policy states what to do with certificates as they are found. In many cases, this difference is moot – the messenger might look in an LDAP directory attached to a CA it considers authoritative. On the other hand, it might also have a local cache of authoritative and merely reasonable certificates.

There are basic mechanisms that the messenger can use as part of its overall trust policy. These relate to types of trust models.

### **3.4.1 Hierarchical Trust**

Hierarchical trust is typical CA trust. A certificate is valid if it descends from a chain of certificates from some trusted root. Most certificate systems, including X.509 and OpenPGP allow for hierarchical trust. For example, a VeriSign Class 3 certificate is valid or not within the context of a hierarchical trust model. A reasonable policy might include that a VeriSign Class 3 certificate is not only valid, but *authoritative*, by which we mean that certificate search stops when finding an authoritative certificate.

### **3.4.2 Cumulative Trust**

Cumulative trust is the typical PGP Web Of Trust. In this model, a certificate is valid if some collection of authorities all agree on a certification. Of all trust models, it is the least directly applicable to a robot messenger, because it relies on human judgment. However, there are two ways it fits neatly into the Self-Assembling PKI.

The most direct is that if the sender uses the web of trust to consider a certificate valid, the messenger can use that human's ruleset to consider a certificate valid or authoritative.

However, if the messenger's policy states that (for example) a VeriSign Class 1 certificate is valid, but not authoritative, this is a form of cumulative trust. The messenger will use that certificate if no better one is found, but it keeps looking for a certificate that is more authoritative.

### **3.4.3 Direct Trust**

Direct trust is extensively used by humans who use OpenPGP-based or S/MIME-based systems, but not very much by machine-driven systems. In direct trust, we consider a certificate to be valid because we got it (or a reference to it) from the entity it represents. For example, many people print their OpenPGP key fingerprint on their business cards. This is a form of distributing direct trust. In other cases, users email each other certificates and trust them based upon that direct transfer of the certificate from user to user.

Years ago, when I met Carl Ellison for the first time, he gave me a business card with a PGP key

fingerprint. Now, nearly a decade later, with many changes of jobs and keys, I still consider his OpenPGP certificates valid based upon the direct trust from that long-lost business card.

Typically, machine-driven systems do not use direct trust, but it is a key mechanism for the Self-Assembling PKI. The messenger uses a number of rules in its policy. One of those rules is to ask the recipient's Internet domain for a suitable certificate.

There are a number of mechanisms that the messenger can use to ask the recipient's domain for a certificate. LDAP directories, DNS, HTTP-based certificate servers, or even queries via SMTP extensions.

Whatever the mechanism, the robot messenger mimics what a human messenger would do. It goes to the address of the recipient and talks to entities there. If the domain has a certificate for the recipient, it may use direct trust to consider the certificate valid or authoritative independent of any other mechanisms.

This policy has obvious limitations. Domains and DNS can be spoofed, although DNSsec, SSL/TLS server certificates, or IPsec certificates can all easily strengthen this. On the other hand, it is relatively easy to use those security mechanisms to enhance the security of cooperating domains such as business partners, so that each entity's PKI transparently and dynamically interoperates with the other one. Furthermore, as these other security systems build the strength of the overall Internet infrastructure, the security of the Self-Assembling PKI increases, and even with no DNSsec etc., it's a vast improvement on sending messages in the clear.

#### **3.4.4 Local Trust Policies**

It is also worth mentioning that any given domain and set of messengers can design their part of the PKI with other small tweaks and policies.

## **4 Look Before You Leap**

Clausewitz said that no battle plan survives contact with the enemy. Similarly, no security policy survives contact with the actual users. Consequently, it is imperative to be able to test policies before they're implemented. A component in the PKI can be left on its own to do things such as create certificates, but before it actually uses them it should be possible to test to see what should be happening.

This is a vital feature of the Self-Assembling PKI which we call "learn mode." Its goal is to speed deployment, and people will deploy slower what they understand less. Consequently, it improves the overall security of the whole system to help administrator understand what they are installing and be able to stay informed while it runs.

Learn mode also permits the first contact a change in policy has with the live network to be benign.

Many policies sound good, but have obvious or inobvious drawbacks. There is perhaps not a person using email who has not thought aloud, “Hmmm, if all incoming email had to be signed, then I wouldn’t be getting all that spam.” Few organizations can actually live with such a policy. Getting a report on exactly how many important messages would have been bounced by such a policy could be an eye-opener. Administrators smile at the thought of being able to produce a report proving just how silly a policy change would be.

## **5 Bootstrapping the PKI**

Here is an example of how a Self-Assembling PKI might be integrated into an existing messaging infrastructure.

Consider an organization, a.com, that installs a Self-Assembling PKI, implemented as proxy server between the domain’s users and their existing email server, using SSL between it and the users.

The proxy server observes connections between the users and the email server as it proxies them. When it sees an authenticated connection, it checks its certificate database for a certificate for that user. If one does not exist, it generates a key and creates a certificate, storing it in the database. The information in that certificate is updated with other information in proxied messages. For example, an authenticated SMTP message contains the common name of the user in message.

Once the user’s certificate has been created, incoming mail for that user can be encrypted. The certificate can also be published in a directory or given to other servers.

This same simple process continues for all the users on this server.

The proxy server also updates the validity dates on the certificates it creates, keeping the ones in use with valid dates.

## **6 Handling Outside Users**

The last problem that any PKI designed for improving deployment needs to address is how to communicate with people who are not part of the PKI at all. Ideally, there are mechanisms in the system to handle this, as it helps the PKI to grow even further.

When the robot messenger cannot find a certificate for the recipient, it has a number of policy options. Least interesting, it can send the message anyway, in plain text. Only slightly less interesting, it could bounce the message back to the sender (this is one of those policies that is tempting, and cries out for being used just for the amusement or education value).

Beyond these two simple, obviously inadequate policies, the messenger has two more sophisticated options:

## 6.1 Smart Trailers

A number of systems that add in security enhancements to messaging, such as automatic virus scanners, frequently add a trailer to the message stating that it has been scanned. Similar to that is a policy that called the smart trailer. A smart trailer says something similar to:

```
-----  
Do you know that email messages are as visible as postcards? This message was  
delivered in the clear for anyone to see, but could have been sent securely.  
If this concerns you, click this url to find out how you can have your email  
delivered to you fully encrypted: <https://sapki.xyz.tld/secure-delivery.html>
```

The specified URL points back to one of the messenger servers, which explains how the PKI works, gives references to available software of all sorts, and provides a form into which a certificate may be placed. This certificate then becomes part of PKI and the next message to that person will be encrypted and formatted accordingly.

## 6.2 Boomerang Mail

Boomerang mail is a procedure for securely delivering a message and attachments to someone outside the PKI. If the messenger's policy calls for a boomerang message, it stores the actual message in a secure spot, and sends to the recipient a separate message similar to this:

```
Date: Mon, 6 Jan 2003 18:25:58 -0800  
From: Jon Callas <jon@pgp.com>  
To: Important Person <vip@host.tld>  
Subject: Important Secure Message
```

Jon Callas would like to send you an important message that should not be delivered in plain text. You may receive this message securely through your web browser. Simply click this link to receive it and follow the directions:

```
<https://boomerang.pgp.com/AVERYLONGURLSYNTHESIZEDFROMASECUREHASHFUNCTION>
```

The URL in the boomerang message allows the recipient to retrieve their message and any attachments. It also has links to the same information that a smart trailer points to, with the same options to supply a certificate, or even merely a password for further boomerang messages. Depending on the enthusiasm of the implementors, it may even have a webmail system with it.

There is, however, one last detail that must be solved. How do recipients authenticate themselves to the messenger to receive their message?

The simplest, yet least secure mechanism is one that we call "*first time good*." This mechanism has no password and simply relies on the fact that the vast majority of email is not read or intercepted

in transit. Like all policies, there are situations where this is adequate, and situations where it is not.

A more secure mechanism not only sends the boomerang message to recipient, but a second message to the sender. This extra message sent to the sender contains a one-time, synthesized password that the recipient will need to retrieve their message. How that password gets to the recipient is left as an exercise for the sender, but phone calls, SMS, and carrier pigeons are all options. We call this “*out-of-band authentication*.”

In either case, however, the recipient can specify a password to use for further messages, or give the messenger a certificate to use in the future. In some cases, the web server may even have downloadable software to further help spread the PKI.

## 7 Risks and Limitations

There are, as mentioned above a number of tradeoffs in this system to achieve its goal of widespread deployment.

- The authentication to the PKI and its systems is relatively weak; typically, it is merely the password that a user normally uses to authenticate the message system. While it’s possible that this could be augmented with stronger authentication, the vast majority authentications are made with nothing stronger than SSL.
- The certificates and keys used by the messengers are held by them, perhaps with the users as well, but are nonetheless not entirely under the users’ control. Depending on the implementation, these may be protected with key-management hardware, but they may be used in a completely software system.
- The certificates in such a PKI are thus necessarily low-valued. Perhaps this is more of an observation than anything more, but the CAs in such a PKI should note that these certificates have a weak semantic meaning. It would be a mistake to use them, in purchases of real estate, for example.
- This system provides no help to its users to assuring they are sending a message to the right person. In the so-called “John Wilson Problem,” an organization has several members all named John Wilson, and they commonly receive each others’ email. The Self-Assembling PKI dutifully, securely delivers mis-addressed messages to the wrong person. It is also vulnerable to weaknesses within the Internet infrastructure such as spoofed domains and hosts.

Nonetheless, in spite of these drawbacks, we are augmenting a system that would otherwise deliver messages in plaintext with the same authentication. There is a risk that the users of the system may believe it to be more secure than it is, but in the world of message security, we are not faced with message systems unimplemented because of security concerns. We are instead faced with message systems that are deployed without security. Even worse, there are many cases of users

migrating to less secure systems such as email on handhelds and instant message systems because of the convenience and utility of these systems. [IM1, IM2]

## **8 Conclusion**

The Self-Assembling PKI is a collection of technologies, strategies, and policies with a goal toward spreading deployment of secure messaging. It also exploits a shift in metaphor from a telephone model to a postal model. This new metaphor gives a new way of examining the issues of creating a PKI that inspires us to create the PKI in such a way that it is easy to use, easy to deploy, and easy to maintain.

## References

- [CHADWICK] Chadwick, D.W. *Deficiencies in LDAP When Used to Support PKI*. Communications of the ACM, March 2003, pages 99–104.
- [GARTNER] Graff, J. The Gartner Group, private communication.
- [GUTMANN] Gutmann, P., *PKI: It's Not Dead, Just Resting* IEEE Computer, August 2002, pages 41–49.
- [IM1] Festa, P. *Business: IM is getting out of control* ZDNet UK, 26 April 2001, <http://news.zdnet.co.uk/story/0,,t269-s2085865,00.html>
- [IM2] Hu, J., *IM: From fad to big business and beyond* CNet News.com, 13 March 2002, <http://zdnet.com.com/2100-1104-992391.html>
- [JOHNNY] Whitten, A., Tygar, J.D., *Why Johnny Can't Encrypt: A Usability Evaluation of PGP 5.0* Usenix Security Symposium, 1999.
- [OPENPGP] Callas, J., Donnerhacke, L., Finney, H., Thayer, R. *OpenPGP Message Format*. RFC 2440, November 1998, <http://www.ietf.org/rfc/rfc2440.txt>
- [PFS] Back, A., Brown, I., Laurie, B., *Forward Secrecy Extensions to OpenPGP* November 2000, <http://www.apache-ssl.org/openpgp-pfs.txt>
- [PGPUSE] PGP Corporation internal survey of its own organizational customers, 2002–2003.
- [RIVEST] Rivest, R., *Can We Eliminate Certificate Revocation Lists?* Proceedings of Financial Cryptography '98; Springer Lecture Notes in Computer Science No. 1465 (Rafael Hirschfeld, ed.), February 1998, pages 178–183.
- [SPKI] Ellison, C., *SPKI Requirements*. RFC 2692, September 1999, <http://www.ietf.org/rfc/rfc2692.txt>
- Ellison, C., Frantz, B., Lampson, B., Rivest, R., *SPKI Certificate Theory*. RFC 2693, September 1999, <http://www.ietf.org/rfc/rfc2693.txt>
- [STUBBLEBINE] Stubblebine, S. *Recent-Secure Authentication: Enforcing Revocation in Distributed Systems* Proceedings of the 1995 IEEE Symposium on Research in Security and Privacy, Oakland, May, 1995, pp. 224–234., <http://www.stubblebine.com/95oak.pdf>