Ten Risks of PKI: What You're not Being Told about Public Key Infrastructure

By Carl Ellison and Bruce Schneier

Computer security has been victim of the "year of the..." syndrome. First it was firewalls, then intrusion detection systems, then VPNs, and now certification authorities (CAs) and public-key infrastructure (PKI). "If you only buy X," the sales pitch goes, "then you will be secure." But reality is never that simple, and that is especially true with PKI.

Certificates provide an attractive business model. They cost almost nothing to make, and if you can convince someone to buy a certificate each year for $5, that times the population of the Internet is a big yearly income. If you can convince someone to purchase a private CA and pay you afee for every certificate he issues, you're also in good shape. It's no wonder so many companies are trying to cash in on this potential market.With that much money at stake, it is also no wonder that almost all the literature and lobbying on the subject is produced by PKI vendors. And this literature leaves some pretty basic questions unanswered: What good are certificates anyway? Are they secure? For what? In this essay, we hope to explore some of those questions.

Security is a chain; it's only as strong as the weakest link. The security of any CA-based system is based on many links and they're not all cryptographic. People are involved.

Does the system aid those people, confuse them or just ignore them? Does it rely inappropriately on the honesty or thoroughness of people? Computer systems are involved. Are those systems secure? These all work together in an overall process. Is the process designed to maximize security or just profit?

Each of these questions can indicate security risks that need to be addressed.

Before we start: "Do we even need a PKI for e-commerce?"

Open any article on PKI in the popular or technical press and you're likely to find the statement that a PKI is desperately needed for e-commerce to flourish. This statement is patently false. E-commerce is already flourishing, and there is no such PKI. Web sites are happy to take your order, whether or not you have a certificate. Still, as with many other false statements, there is a related true statement: commercial PKI desperately needs e-commerce in order to flourish. In other words, PKI startups need the claim of being essential to e-commerce in order to get investors.

There are risks in believing this popular falsehood. The immediate risk is on the part of investors. The security risks are borne by anyone who decides to actually use the product of a commercial PKI.

Risk #1: "Who do we trust, and for what?"

There's a risk from an imprecise use of the word "trust." A CA is often defined as "trusted."

In the cryptographic literature, this only means that it handles its own private keys well. This doesn't mean you can necessarily trust a certificate from that CA for a particular purpose: making a micropayment or signing a million-dollar purchase order.

Who gave the CA the authority to grant such authorizations? Who made it trusted?

A CA can do a superb job of writing a detailed Certificate Practice Statement, or CPS — all the ones we've read disclaim all liability and any meaning to the certificate — and then do a great job following that CPS, but that doesn't mean you can trust a certificate for your application. Many CAs sidestep the question of having no authority to delegate authorizations by

issuing ID certificates. Anyone can assign names. We each do that all the time. This leaves the risk in the hands of the verifier of the certificate, if he uses an ID certificate as if it implied some kind of authorization.

There are those who even try to induce a PKI customer to do just that. Their logic goes: (1) you have an ID certificate, (2) that gives you the keyholder's name, (3) that means you know who the keyholder is, (4) that's what you needed to know. Of course, that's not what you needed to know. In addition, the logical links from 1 to 2, 2 to 3 and 3 to 4 are individually flawed. [We leave finding those as an exercise for the reader.]

Long keys don't make up for an insecure system because total security is weaker than the weakest component in the system. The same applies to the verifying computer—the one that uses the certificate.

Risk #2: "Who is using my key?"

One of the biggest risks in any CA-based system is with your own private signing key. How do you protect it? You almost certainly don't own a secure computing system with physical access controls, TEMPEST shielding, "air wall" network security, and other protections; you store your private key on a conventional computer. There, it's subject to attack by viruses and other malicious programs. Even if your private key is safe on your computer, is your computer in a locked room, with video surveillance, so that you know no one but you ever uses it? If it's protected by a password, how hard is it to guess that password? If your key is stored on a smart card, how attack-resistant is the card? [Most are very weak.] If it is stored in a truly attack-resistant device, can an infected driving computer get the trustworthy device to sign something you didn't intend to sign?

This matters mostly because of the term "non-repudiation." Like "trusted," this term is taken from the literature of academic cryptography. There it means something very specific: that the digital-signature algorithm is not breakable, so a third party cannot forge your signature. PKI vendors have latched onto the term and used it in a legal sense, lobbying for laws to the effect that if someone uses your private signing key, then you are not allowed to repudiate the signature. In other words, under some digital signature laws (e.g., Utah and Washington), if your signing key has been certified by an approved CA, then you are responsible for whatever that private key does. It does not matter who was at the computer keyboard or what virus did the signing; you are legally responsible. Contrast this with the practice regarding credit cards. Under mail-order/telephone-order (MOTO) rules, if you object to a line item on your credit card bill, you have the right to repudiate it — to say you didn't buy that — and the merchant is required to prove that you did.

Risk #3: "How secure is the verifying computer?"

The previous section showed that the computer holding or driving the private key needs to be secure. Long keys don't make up for an insecure system because total security is weaker than the weakest component in the system. The same applies to the verifying computer - the one that uses the certificate.

Certificate verification does not use a secret key, only public keys. Therefore, there are no secrets to protect. However, it does use one or more "root" public keys. If the attacker can add his own public key to that list, then he can

issue his own certificates, which will be treated exactly like the legitimate certificates. They can even match legitimate certificates in every other field except that they would contain a public key of the attacker instead of the correct one.

It doesn't help to hold these root keys in "root certificates." Such a certificate is self-signed and offers no increased security. The only answer is to do all certificate verification on a computer system that is invulnerable to penetration by hostile code or to physical tampering.

Risk #4: "Which John Robinson is he?"

Certificates generally associate a public key with a name, but few people talk about how useful that association is. Imagine that you receive the certificate of John Robinson. You may know only one John Robinson personally, but how many does the CA know? How do you find out if the particular John Robinson certificate you received is your friend's certificate? You could have received his public key in person or verified it in person (PGP allows this), but more likely you received a certificate in e-mail and are simply trusting that it is the correct John Robinson. The certificate's Common Name will probably be extended with some other information, in order to make it unique among names issued by that one CA. Do you know that other information about your friend? Do you know what CA his certificate should come from?

When Diffie and Hellman introduced public-key cryptography, they proposed a modified telephone directory in which you could find public keys. Instead of name, address, and phone number, it would have name, address, and public key. If you wanted to find John Robinson's public key you would look him up in the directory, get his public key and send him a message for his eyes only using that public key. This might have worked with the Stanford Computer Science Department phone directory in 1976, but how many John Robinsons are in the New York City phone book, much less in a hypothetical phone book for the global Internet?

It doesn't help to hold these root keys in "root certificates." Such a certificate is self-signed and offers no increased security. The only answer is to do all certificate verification on a computer system that is invulnerable to penetration by hostile code or to physical tampering.

We grow up in small families where names work as identifiers. By the time we're 5 years old, we know that lesson. Names work. That is false in the bigger world, but things we learn as toddlers we never forget. In this case, we need to think carefully about names and not blindly accept their value by the 5-year-old's lessons locked into our memories.

Risk #5: "Is the CA an authority?"

The CA may be an authority on making certificates, but is it an authority on what the certificate contains? For example, an SSL server certificate contains two pieces of data of potential security interest: the name of the keyholder (usually a corporate name) and the DNS name for the server. There are authorities on DNS name assignments, but none of the SSL CAs listed in the popular browsers is such an authority. That means that the DNS name in the certificate is not an authoritative statement. There are authorities on corporate names. These names need to be registered when one gets a business license. However, none of the SSL CAs listed in the browsers is such an authority. In addition, when some server holds

an SSL server certificate, it has permission to do SSL. Who granted the authority to an SSL CA to control that permission? Is the control of that permission even necessary? It serves an economic purpose (generating an income stream for CAs) but does it serve a security purpose? What harm is done if an uncertified server were allowed to use encryption? None.

Risk #6: "Is the user part of the security design?"

Does the application using certificates take the user into account or does it concern itself only with cryptography?

For example, a normal user makes a decision of whether to shop with a given SSL-protected Web page based on what is displayed on that page. The certificate is not displayed and does not necessarily have a relation to what is displayed. SSL security does not have the ability to control or even react to the content of the Web page, only its DNS address. The corporate name is not compared to anything the user sees and there are some Web pages whose certificate is for a company that does Web hosting, not for the company whose logo appears on the displayed page. Users can't, and can't be expected to, sort this all out.

Risk #7: "Was it one CA or a CA plus a Registration Authority?"

Some CAs, in response to the fact that they are not authorities on the certificate contents, have created a two-part certification structure: a Registration Authority (RA), run by the authority on the contents, in secure communication with the CA that just issues certificates. Other vendors sell CA machinery directly to the content authority.

The RA+CA model is categorically less secure than a system with a CA at the authority's desk. The RA+CA model allows some entity (the CA) that is not an authority on the contents to forge a certificate with that contents. Of course, the CA would sign a contract promising not to do so, but that does not remove the capability. Meanwhile, since security of a chain is weaker than the weakest link, the RA+CA is less secure than either the RA or the CA, no matter how strong the CA or how good the contract with the CA. Of course, the model with a CA at the authority's desk (not at the vendor's site) violates some PKI vendors' business models. It's harder to charge for certificates when you sell someone the CA code (or they get it for free, as Open Source).

Meanwhile, since security of a chain is weaker than the weakest link, the RA+CA is less secure than either the RA or the CA, no matter how strong the CA or how good the contract with the CA.

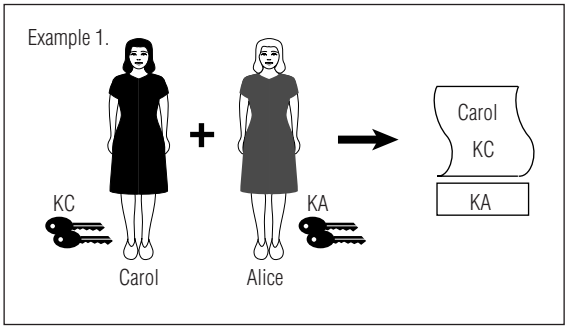
Risk #8: "How did the CA identify the certificate holder?"

Whether a certificate holds just an identifier or some specific authorization, the CA needs to identify the applicant before issuing the certificate.

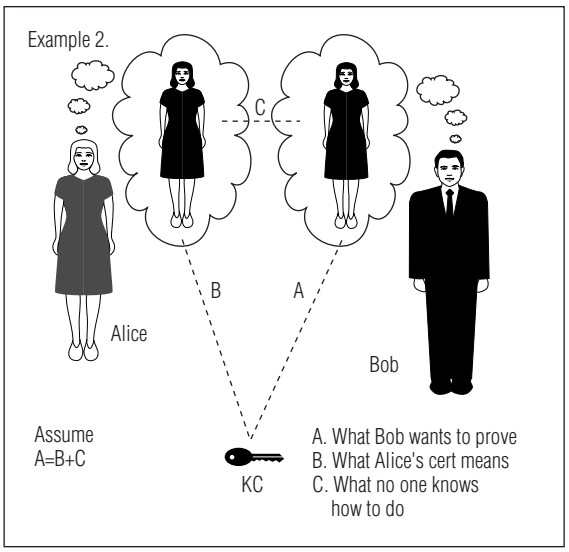
There was a credit bureau that thought they would get into the CA business. After all, they had a vast database on people, so, the thinking ran, they should be able to establish someone's identity online with ease. If you want to establish identity online, you can do that provided you have a shared secret with the subject and a secure channel over which to reveal that secret. SSL provides the secure channel.

The trouble with a credit bureau serving this role is that in their vast database there is not one secret shared with the subject. This is because credit bureaus are in the business of sell-

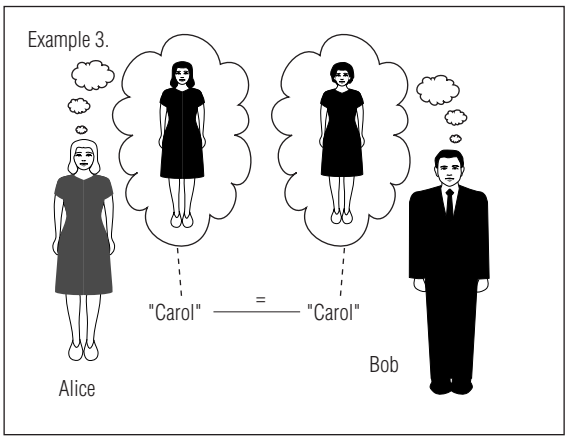
Problems of Authentication



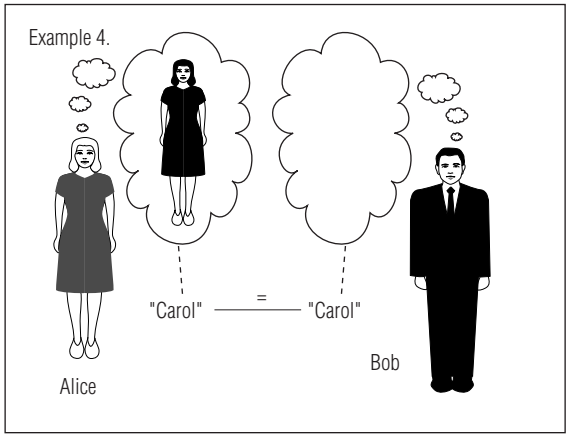
In this example, there is no problem. Alice meets Carol and verifies her identity. Carol demonstrates that she controls key Kc.



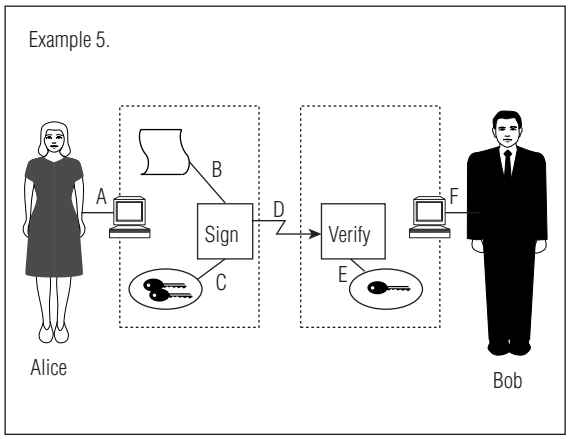
In this example, Bob wants to know who is using the key Kc. But there is no way he can meet Carol, as Alice did in Example 1. Bob must settle on finding out if Carol was assigned the key Kc. But to do so, Bob must establish the relationsip A, by using relationships B and C. Relationship B is Alice's certificate. Relationship C is some comparison of mental images. But nobody knows how to do that.



In this example, Bob and Alice compare names instead of mental images. There problem here is that they could both be using the name "Carol" for different people without knowing it–because the names compare correctly.



In this example, Bob has no mental image of Carol, so the name "Carol" doesn't tell Bob anything.



Keeping in mind the problems illustrated in Examples 2 thru 4, consider the risks that arise when Bob and Alice want to communicate. [[Display the following as call-outs corresponding to points of attack illustrated in art work]]
A) Someone could launch an attack using Alice's computer.
B) Someone could display one document to Alice to get her approval for signature, then send a different document to be signed.
C) Someone could steal Alice's keys, from inside her computer or steal the passphrase for her keys.
D) Someone could attempt to attack the crypto-protected channel (but it is probably not worth attacking, since there are so many other options).
E) Someone could replace Alice's key with their own. If Alice's key is protected by a certificate, the attacker could replace the certificate root key and issue new certificates for their own key.
F) Someone could lie to Bob about whether the signature was verified.
Caption continued: The point is that there are many links in the path that are not protected by cryptography. These links can be subverted to get Bob to accept something that is not really from Alice or at least not what she intended.

ing their information to people other than the subject. Worse, because credit bureaus do such a good job at collecting and selling facts about people, others who might have information about a subject are probably hard pressed to find any datum shared with the subject that is not already available through some credit bureau. This puts at risk commercial CAs that use credit bureau information to verify identity online; the model just doesn't work.

Meanwhile, having identified the applicant somehow, how did the CA verify that the applicant really controlled the private key corresponding to the public key being certified? Some CAs don't even consider that to be part of the application process. Others might demand that the applicant sign some challenge right there on the spot, while the CA watches.

Certificates must be used properly if you want security. Are these practices designed with solid security reasons, or are they just rituals or imitations of the behavior of someone else? Many such practices and even parts of some standards are just imitations which, when carefully traced back, started out as arbitrary choices by people who didn't try to get a real answer.

Risk #9: "How secure are the certificate practices?"

Certificates aren't like some magic security elixir, where you can just add a drop to your system and it will become secure. Certificates must be used properly if you want security. Are these practices designed with solid security reasons, or are they just rituals or imitations of the behavior of someone else? Many such practices and even parts of some standards are just imitations which, when carefully traced back, started out as arbitrary choices by people who didn't try to get a real answer.

How is key lifetime computed? Does the vendor use 1 year, just because that's common? A key has a cryptographic lifetime. It also has a theft lifetime, as a function of the vulnerability of the subsystem storing it, the rate of physical and network exposure, attractiveness of the key to an attacker, etc. From these, one can compute the probability of loss of key as a function of time and usage. Does the vendor do that computation? What probability threshold is used to consider a key invalid?

Does the vendor support certificate or key revocation? Certificate Revocation Lists (CRLs) are built into some certificate standards, but many implementations avoid them because they seem to be archaic remnants of the newsprint booklets of bad checking account numbers one used to find at the supermarket checkout stand. Like those booklets, CRLs are seen as too big and too outdated to be relevant. However, if CRLs are not used, how is revocation handled?

If revocation is handled, how is compromise of a key detected in order to trigger that revocation? Can revocation be retroactive? That is, can a certificate holder deny having made some signature in the past? If so, are signatures dated so that one knows good signatures from suspect ones? Is that dating done by a secure timestamp service?

How long are the generated public keys and why was that length chosen? Does the vendor support 512-bit RSA keys just because they're fast or 2048-bit keys because someone over there in the corner said he thought it was secure?

Does the proper use of these certificates require user actions? Do users perform those actions? For example, when you establish an SSL connection with your browser, there's a visual

indication that the SSL protocol worked and the link is encrypted. But who are you talking securely with? Unless you take the time to read the certificate that you received, you don't know. Even then, you may not know (cf., Risk #4, above) but if you don't even look, it's much like going into a private room with the lights off: you might know that someone else is there and your conversation is private, but until you know who that other person is, you shouldn't reveal any secret information.

Risk #10: "Why are we using the CA process, anyway?"

One PKI vendor employee confided in us a few years ago that they had great success selling their PKI solution, but that customers were still unhappy. After the CA was installed and all employees had been issued certificates, the customer turned to the PKI vendor and asked, "OK, how do we do single sign-on?" The answer was, "You don't. That requires a massive change in the underlying system software."

Single Sign-On (SSO) might be the killer app of PKI. Under SSO, you come into work in the morning, plug in your smart-card, enter the PIN that activates it, and for the rest of the day, you don't have to do any more logins. All of that is handled for you by the SSO mechanism.

Attractive isn't it? Of course, it's attractive. Authentication is a pain. Anything we can do to avoid it, we'll jump at.

Unfortunately, the security value of authentication is all but completely defeated by SSO. Authentication is supposed to prove that the user is present at the controlling computer, at the time of the test. Under SSO, when the user has to rush to the washroom, any passing person can walk up to that user's computer and sign on someplace via the SSO mechanism.

So, why are so many jumping at the CA process with such fervor? Do they use certificates out of empty ritual, just because the other guy does and it's the thing to do this year? Do they do it in order to pass the liability buck: to be able to blame the PKI experts if any insecurity sneaks through?

We are not that cynical. Our assessment is that security is very difficult, both to understand and to implement. Busy system administrators and IT managers don't have the time to really understand security. They read the trade press. The trade press, influenced by PKI vendors, sings the praises of PKIs. And PKI vendors know what busy people need: a minimal-impact solution. "Here, buy this one thing and it will make you secure." So that's what they offer. Reality falls far short of this promise, but then, this is a business and the prominent voices are those with something to sell. Caveat emptor.

Bruce Schneier is the author of Applied Cryptography, the Blowfish and Twofish encryption algorithms, and dozens of research papers and articles on cryptography and computer security. He is CTO of Counterpane Internet Security, Inc., a managed security service company offering leading-edge expertise in the fields of intrusion detection and prevention, preemptive threat discovery, forensic research, and organizational IT systems analysis. You can subscribe to his free monthly e-mail newsletter, Crypto-Gram, at http://www.counterpane.com

Carl M. Ellison is a Senior Security Architect for Intel Corporation, with special focus on cryptography, cryptographic access control and public key certificates. Prior to the focus on cryptography, his earlier professional computer science career focused on system design with special emphasis on distributed and networked systems.