PGP for companies, why not?

Rick van Rein
OpenFortress Digital signatures
rick@openfortress.nl

July 15, 2005

Abstract

There are good reasons why PGP is more suitable for commercial applications than X.509, but also several reasons that make it less usable. We will compare the technologies to find out the things that block PGP from commercial applications.

Introduction

There are currently two major signing technologies for general use, namely PGP en X.509 – the first based on a web of trust with peer-to-peer trust relationships, and the second based on a hierarchy of trust relations, with an ultimately trusted root signer.

In everyday practice, X.509 is used with small *islands of trust*, such as the certificate hierarchy employed by a lot of companies that rely on active directory and related Windows-based solutions. Such certificates are internally useful, but not across company boundaries. The practical use of X.509 between companies is generally limited because there is no common infrastructure to exchange the internal root certificates; and the globally operating root certificates that come pre-installed with browsers don't seem to get a world-wide X.509-based PKI going either, except for trivial applications like TLS server certificates.

The PGP system is not based on a *central trust provider* such as is needed for X.509 systems. This is because PGP grows from locally made signatures, which integrate in its web of trust. This means that a web of trust grows bottom-up instead of under a heavy top. Commercial signers often prefer to be at that top, because the strict hierarchy of X.509 implies a *lock-in* under the root key of the certificate authority; PGP does not have such commercially advantageous constructions.

But let's forget about the viewpoint of certificate signers — what is in the interest of users of a signing technology? And how does the viewpoint of a signature verifier differ from that of a signer?

1 Digital signing practice of today

Source: http://openfortress.nl/news/security/sign-in-browser.2005-01-08-22-22.article – Digital signing in a browser? No thanks!

Source: http://openfortress.nl/news/security/sign-plaintext.2005-01-08-23-58.article — Be careful what you sign.

Current signing applications often focus on the browser as the everybody-can-use-it interface to their programs. Popular as browsers may be, it is a very, very bad idea to conduct signing in the browser, especially for those who cannot be reached through other means.

If you let people sign for a document, they'd better know what they are doing. A process that is guided externally, that is by the site that tries to pull a signed statement out of someone, is not exactly the best process possible. It is made as simple as possible, but at the expense of *taking control* over the signature-placing. The browser dictates what happens, and when. There is no room for initiative, and no room for really understanding what is going on.

A big problem with guiding an un-knowing customer through a signing process is that it is going to be exceptionally hard to make it stand in court. The certainties of cryptographic technology may be hard as rock, but legal processes tend to support common sense. And common sense is likely to reason that signatures placed unknowingly are no legally acceptable signatures.

Related to this problem is the tendency to get the signature on HTML documents, or semantically even less defined, XML documents. The problem with these formats is simply that they have something to hide, namely their source code. As demonstrated on the second source above, it is not hard to make a statement that varies to the liking of the composer of the document.

Ideally, signatures should be placed on documents which have nothing to hide. In other words, the source code should be intelligible to the reader. And if source code must be readable, there is only one serious option, namely plain old ASCII text. It doesn't look as fancy as something like a LaTeX document, but at least it bypasses the need to decipher the author's TeX hacks! As soon as you prefer readable source over layout, you end up with plain text.

2 The Meaning of a Signature

Source: http://openfortress.nl/news/projects/sigpolicy/ — OpenFortress Project: Signing policies.

Signatures can have many different meanings. If I buy a house, I make a strong statement by signing the buying contract. If I receive a parcel by mail, I may sign much weaker, only to indicate having received it. If I talk to a lawyer, I may be inclined to sign so they cannot rip parts of my words out of their context and ask if I signed that.

In electronic signing, there is a shortage of this degree of subtlety. Signing technologies tend to support signing policies, at the very least in certification signatures, but this technical facility is rarely used. And the reason is clear: communication of such legal/English texts is not going to simplify our daily lives.

Signing policies are used mainly by signing authorities, and perhaps a few souls who want to make a strong statement about the meaning of their PGP

key signatures. But they are not practically useful because it is intended for human interpretation, and because it lacks a consistent structure.

The idea that we propose is to not use a URL for a signing policy, but use a URN instead. This is possible in signing technologies that refer to URIs, which holds for PGP, X.509 and XML signing. URNs define name spaces that can have a syntax that could be automatically parsable, if well-defined. OpenFortress is working towards a library, to be released under at least the GPL, that can be used to evaluate such signing policies automatically.

This approach would be of great interest to express the intentions of a signature; but it should not be done in such a manner that the largest party (signer or verifier) dictates the policy. Instead, there should be a widely adopted set of *policy modules* that can individually be accepted or rejected by both parties. Such modules should be composable units with a textual component (to support their evaluation by humans) and an automatically processable part.

Important for such modules is to come to a set of modules that are generally thought useful. For example, a module idchecked and another idcopyfiled could be composed at will. Such modules could be used to compare certification authorities objectively. All that is needed is a centrally coordinated effort to identify and construct such modules. This should be done in a sort of *general consensus* process that is open to all participants who are interested. This means that it is probably a good idea to standardise such modules through RFCs that are written under the IETF consensus process. IANA could then file the identifiers and ensure their uniqueness.

The same technology can be used for one-to-one or one-to-many relationships with a very specific agreement. A locally usable module notation like x-sha1=12345... could facilitate such situations. The RFC-based modules are probably more suitable for open, many-to-many relationships.

The advantage of using signing policies for this purpose is that it evades the need for escapes in the text to be signed, which is otherwise needed to insert automation-supportive constructions in plain text. The policies can be handled by end-user applications instead of being presented as part of the text to be signed. Users who sign under a policy are likely to group accepted modules under a profile – one profile for gossip, another for electronic banking.

Note that these constructs are possible with the common signing technologies — PGP has built-in support; X.509 has complete built-in support for certificate signatures and can be enhanced with an attribute in the CMS standard; XML signatures can incorporate references to URIs including URNs. The only problem is that not all technologies make these policies critical — if they don't recognise it they may silently skip it, which may cause (legal) problems.

Given signing policies, there is much to be gained. The PGP community can sign equivalence between keys of the same owner, to support moving to a newer key without adding one to the distance to signers on the old key. Reputation systems can be implemented using PGP, both for companies (we have traded with them since 1987) and for persons (I know him so well). Large databases can support emails with editing statements, provided that these are signed with acceptance of their policies. And these are just a few examples.

3 A commercial perspective on PGP

Commercial applications of signing should, first and foremost, be practical. This is a strong point in favour of PGP. X.509 systems are more complex to setup and deploy than PGP systems. Also note that companies tend to think of trust in peer-to-peer terms, and to easily go across borders. This means that PGP is a much better fit to the needs of a company than X.509. Put bluntly, X.509 is mainly of interest to certification providers, who prefer to be the only signer instead of one possible signing partner.

Why then, is it that companies rarely rely on PGP? An important problem with PGP is its loose structure. There are no certainties to be drawn from a signature in the PGP system, because the best guarantee possible is *carefully checked*. What does that mean exactly? Are we speaking of *confirmation of reputation* or are we speaking of *passport-based identification*? Without certainty, there is no basis for gaining trust and thus no basis for trading.

A second problem with PGP for commerce is, like it or not, its web of trust. Companies prefer not to rely on intermediates, and if they exist there should be at most one to make it possible to complain, ask questions, and demand corrections or compensations. The web of trust was not designed for this perspective, but more to protect the signers who are people from everywhere on the world. There may however be a subset willing to make stronger statements and give companies the basis for trust they need – and signing policies can be made to allow transitive trust acceptable.

Finally, an important problem surrounding PGP is its *end user complexity*. End users are assumed to be understand things as confusing as a correct signature from an untrusted source, they are assumed to validate relationships with others, and assign a level of trust to them. All this could be centrally coordinated for a company, but only when automation is used wherever possible. The aforementioned signing policies in URN-form can help to setup a minimum boundary for signatures on incoming mail in an automatic way. OpenFortress is currently developing such a filtering tool.

Conclusion

PGP has a more suitable structure for commercial applications than X.509, but the way that it spreads trust is unsuitable for commercial utilisation. We believe that automatically processable signing policies with a set of predefined policy modules can solve most problems that commercial companies may have with PGP.

For more information on this project, we invite you to our website, notably to http://openfortress.nl/news/projects/sigpolicy/