

Simple Public Key Infrastructure

Eloi Sanfèlix González

Abstract

This paper summarizes the work done by the IETF SPKI working group, which defines a *simple* public key infrastructure focused on authorization rather than authentication. The paper looks at the specification analyzing and criticizing its new contributions.

1 Introduction

Simple Public Key Infrastructure (SPKI) is defined by a set of four documents published by the IETF. An Internet draft ([DRAFT1]) provides a set of examples where a SPKI could be used; draft [DRAFT2] provides a formal definition of the SPKI certificates, from the format of the certificate to the reduction of certificates to a so called *tuple* containing some information about the holder of the certificate.

In turn, [RFC2692] explains the security requirements of SPKI and possible use cases obtained by public discussion. Finally, [RFC2693] explains the theory behind SPKI certificates and access control lists (ACLs) without technical details about the structures and covering a wide range of aspects, from name certification to key management.

Therefore, these 4 documents are addressed to different people profiles: developers implementing SPKI into its products, people deploying SPKI systems, and anyone curious about the structure and functioning of SPKI. The following sections present a walk through the SPKI specifications.

2 Security requirements

As mentioned above, the security requirements for SPKI are defined in [RFC2692]. The main security requirement for SPKI is authorization rather than authentication, i.e. a SPKI certificate aims to give certain permissions to the keyholder instead of certifying that the keyholder is the one he claims to be.

It is also a requirement that certificates carry as less information as possible in order to prevent privacy violations, reducing that information to what is really needed to show that a keyholder is authorized for a given action. The system should also support anonymity and blinded signatures, because they are required for certain applications.

Revocation of certificates should be also supported, in form of revocation lists (CRLs) and online one time tests, which assess the validity of

the key at a given time.

Finally, the certificate format must be as simple as possible, in order to allow a wide variety of developers to implement them in its code without relying on third party libraries and to be able to use them in environment with limited resources.

3 The problems of name certification

The SPKI working group introduced the idea that name certificates and more concretely X.509 certificates are inherently flawed, because even though humans are used to identify each other by means of names, these names are normally used in a local scope; however, machines do not need to identify a keyholder by its name, and they need to translate this name into some useful information by consulting data bases or by other mechanisms.

For such a translation, a name should uniquely identify a keyholder, and locally defined names are not useful anymore: one needs global identifiers in order to achieve a one-to-one mapping between names and the related information needed. By these reason, SPKI defends the use of key pairs as a global identifier source; instead of having bindings between keys and names, they propose to bind keys with this related information machines need, as can be seen in the next section.

4 SPKI certificate

SPKI defines the *authorization certificate* as its basic certificate form. This certificate joins a given subject to some authorization information, which in its most basic form means linking a public key together with a specific permission. Furthermore, the subject can be set to a name, a keyholder, an object like a web page or a program and even a *threshold object* specifying that the permission can only be granted to a group of k out of N subjects.

The inclusion of names introduces the need for mapping these names to public keys, in order to make sure that someone presenting a given authorization certificate issued for a name is really the authorized person. This also requires some kind of naming authority being able to certify the relation between names and public keys and a procedure to reduce this information to a $\langle public\ key, authorization \rangle$ certificate.

The decision taken with respect to name certificates is that any certification authority can issue a name certificate but these should not be taken as a global identifier. The naming standard used is SDSI (see [SDSI]), which provides a way to use local names in a global scope by means of name chaining using the name and the name space where it was defined.

It is also possible to have named groups, by means of having multiple name certificates with the same name and different keys. This provides the ability to give certain permissions to a group of individuals using a local $\langle name, authorization \rangle$ pair, identified by the group name and linked

together with these names certificates.

Furthermore, a SPKI certificate can provide the ability to delegate the granted authorization to other subjects. This is done by means of a field in the certificate stating whether this authorization can be delegated by this subject or not. Even if the authorization can be delegated, the certificate holder is still able to execute these permissions, which actually makes sense because it would always be possible to generate a new key pair and delegate the permissions to this new key pair.

In [RFC2693] some arguments are given for the choice of this delegation model, and also for some other models that were considered. In fact, it is possible to argue in favor of all the three delegation models proposed: assuming key holders are honest and do not publish their private keys, it would be desirable to have control on the depth of the delegation chain so that after some delegations it is not possible to further pass the permission. But it is also true that a key holder could publish its own private key if delegation is forbidden, so that no delegation is needed to give authorization. This is the argument given by the defenders of the *no control* delegation model, as opposed to the *integer depth control* and the *boolean depth control* models.

The choice of a boolean scheme where delegation can or cannot be passed leaves the choice of giving permission to further delegate an authorization to the party issuing the delegation. Therefore, it would be possible to forbid further delegation at a given point and assuming honest parties one would achieve something similar to a depth control without the need to add extra complexity to the processing of the authorization information.

This is a trade off between the given solutions, where some control can still be applied assuming that most of the users will give enough value to their private keys to keep them secret, while allowing trusted parties to pass the permissions they have been given.

Additionally, the validity conditions of an SPKI authorization certificate are also defined in [RFC2693]. As usual, the certificates carry validity dates defining the time range in which this certificate is valid under *normal* conditions. Furthermore, three different ways of refining the validity via online tests are defined:

Certificate Revocation Lists: These are listings including the certificates that have been revoked, and therefore are invalid. SPKI defines a deterministic verification using CRLs, opposed to the early model where each CRL had a sequence number and a newer CRL replaced an older one. In SPKI, a certificate susceptible of being referenced in a CRL should include a reference to the key pair that will sign the CRL, the CRL must carry validity dates, and these date ranges must not intersect so that there is one and only one valid CRL at a given moment. The certificate might also include one or more locations where the corresponding CRL can be fetched. Following these rules and requiring the access to the corresponding CRL in every use of the certificate makes the process deterministic, and prevents an active attacker from avoiding the test of the CRL

by means of network attacks such as a Denial of Services.

Revalidation: This mechanism is a positive statement, meaning that a revalidation marks a given certificate as a valid one, instead of an invalid one like CRLs do. The revalidation process is required to follow the same rules stated for CRLs, in order to be a deterministic process.

One-time revalidations: These tests provide a method for querying for the validity of a given certificate whose lifetime is limited to the current authorization computation. Therefore, this offers really fine-grained control over the validity of a certificate. In order to prevent replay attacks, a nonce is included in the request and returned in the signed response.

Furthermore, an example showing how to perform an authorization computation is provided. This is done by means of converting the certificates involved into an intermediate form by means of a signature verification and possibly other work like checking CRLs, online one-time tests, etcetera . These intermediate forms called *tuples* are supposed to be stored in secure memory and hold all the information needed to take the authorization decision.

The reduction process from certificates to *tuples* depends on the kind of certificate provided. For instance, if it is an SPKI certificate, it is mapped directly to one of the existing intermediate forms; on the other hand, a prover could provide a different kind of certificate (e.g. an X.509 certificate or a PGP certificate) and a translation process should be performed in order to achieve the reduction. As already explained, SPKI also defines name certificates for human convenience, which provide the usual key-name binding, and should be reduced to a public key before the authorization computation. For this case, a special intermediate form is defined because the certificate format is substantially different from the normal SPKI authorization certificate.

Finally, the RFC also defines the possibility that an authorization computation results in a new certificate, when the authorization request is asking what a given subject is allowed to do, instead of asking whether a given permission is granted to the subject.

5 Conclusions

The work of the SPKI working group mainly contributes a new kind of certificates linking together authorization information and public keys instead of identities and public keys. The reasons for doing so are that global names (or identities) are difficult to obtain and computers may need to take decisions based on the identification information a certificate carries. Instead of relying on the names, SPKI relies on public keys and makes life simpler to people developing authorization systems based on PKI.

Even though the initial intention was to have a certificate as simple as possible, the final result is not that simple and reading through the

documentation may turn out to be a bit confusing at the beginning. In any case, the resulting format is still simpler than X.509 certificates with their vast amount of possible extensions and does the job for authorization systems.

Therefore, SPKI provides a nice framework for authorization systems based on PKI supporting a wide variety of functions such as threshold authorizations, named groups and local name to key mappings without relying on the assumption that there exist globally unique names and in a simpler way than other alternatives.

Nowadays the specification is in a mature state, and some products and auxiliary tools supporting SPKI have been developed. For a list of these products, research papers and related work one can visit the SPKI web site by Carl M. Ellison at <http://theworld.com/~cme/html/spki.html>.

References

- [DRAFT1] Internet Draft - *SPKI Examples*
- [DRAFT2] Internet Draft - *Simple Public Key Certificate*
- [RFC2692] RFC 2692 - *SPKI Requirements*
- [RFC2693] RFC 2693 - *SPKI Certificate Theory*
- [SDSI] R. Rivest and B.Lampson, *SDSI - A simple distributed security infrastructure* , <http://theory.lcs.mit.edu/~cis/sdsi.html>