Evaluating Revocation Management in SPKI from a User's Point of View

Kristiina Karvonen[1, 2]***, Yki Kortesniemi***[1]***, Antti Latva-Koivisto***[1]
[1]*Helsinki University of Technology, P.O. Box 9800, 02015 HUT, Finland*
[2]*Visual Systems/ TietoEnator Ltd, Vanha Talvitie 19 a, 00580 Helsinki, Finland*
{Kristiina.Karvonen,Yki.Kortesniemi,Antti.Latva-Koivisto}@hut.fi

Abstract

The topic of computer and network security has gained an ever-increasing amount of interest in recent years. The pervasiveness of computers everywhere means that novel users, from novice to expert, need to be able to manage their own security in an understandable way, when giving information about themselves or making transactions online. In this paper, we will present, discuss, and analyse the various revocation methods of a Simple Public Key Infrastructure (SPKI) certificate based access control mechanism from a user's point-of-view. We will consider the downsides and benefits of each revocation method, and make recommendations for which methods to use in which use situations, and how to present the best choices to the user in an understandable way.

Key words: revocation, management, authorisation, certificates, usability

1. Introduction

Controlling access to valuable resources is a necessity, be they traditional resources such as a safe or a bank account, or computer resources, such as a file containing secret information. Managing the access can be a challenging task for the system designers and resource owners, particularly if the system is large and distributed. Currently, there are many kinds of security architectures used to manage the security of these systems. With most, the trend is to involve the users more in handling their security. The users have to decide whom to trust, to what extent, where and when.

Traditionally, access control systems have been based on the concept of Access Control List (ACL), where every resource is bundled with a list of authorised users. Typically, the list is located next to the resource, with all the relevant information in one place. An example could be the VIP list at the door of a club. If there are several doors, we need several copies of the list, and keeping them up to date requires extra work. Authorisation certificates reverse the concept, turning the centralised system into a distributed one. With certificate-based systems, the users of the resource are given a ticket that proves they have the right to use the system. The right no longer resides with the resource but with the user (all the VIPs have a special card they show at the door), and the right automatically follows the users to whichever copy of the resource they go to. If the club has several entrances, the VIP can use any one of them. A significant difference to the paper-based tickets is that authorisation certificates can be used to delegate the rights to other users without any help from the owner of the resource: users can delegate their own rights. This means that it becomes possible e.g. to create new credit cards that make it possible for children to use their parents credit right in such a way that the parents keep their own card and the children have a limit to the amount they can charge from the card [Heikkilä-Laukka, 1999].

Certificates are intuitive in many ways – a certificate granted means a right granted – but sometimes there is a need to cancel, to revoke, a right previously granted. Unfortunately, the revocation mechanisms create problems from the user's point-of-view, because revoking a right is not as intuitive as granting a right, and because there are many mechanisms of revocation to choose from. The core of this paper is to pinpoint and answer the specific usability issues that these revocation mechanisms give rise to, and to be able to choose between them. We will do this by first having a look at existing research on usability of computer security – the area the work at hand falls into, describe the various revocation methods in SPKI, and outline the usability issues that these revocation methods give rise to with the help of several use cases. We will conclude with bringing together the outcome of the analyses of these use cases and their significance for the usability of revocation management in SPKI.

2. Usability of Security - Previous Work

Usability issues in revocation management are part of the problems in usability of computer security, still taking its first steps. It is a generally known fact that users are often considered to be the "weakest link", when computer security issues are at hand ([e.g., Adams and Sasse]). Rightly so, for no matter how sophisticated security mechanisms we use, they are only effective when used correctly. However, more and more "ordinary" people, without any former experience with security issues or technologies will have to learn to manage security now, and even more so in the future. One answer might be to increase the automation level of security. In this scenario, security would be taken care of by the system on behalf of the user, and the user need not bother about it. However, Whitten and Tygar [1999] state that even though automation may be the right solution for securing the communication channel itself, there remain situations where automation is not and cannot be the answer. At many points manual involvement is required from the user, for example when giving access to shared files for others. They also argue that usability of security has some specific usability problems not encountered in other areas. These include making users aware of the security tasks at hand, providing guidance throughout the procedure, and preventing dangerous errors [Whitten-Tygar, 1999]. Adams and Sasse [1999] give more or less the same recommendations for creating usable security in their treatment on how to make passwords user-friendlier, such as motivating users and providing feedback.

A further problem with usability of security is to define the right level of information provided. In case of certificate revocation, it seems we have two options: either we can hide the certificates from the users as fully as possible, or we must make the certificates understandable from the user's point-of-view. A mixture of hiding and revealing information about the certificates, along with preventing the user from making any serious mistakes, is a likely solution.

3. Revoking SPKI Authorisation Certificates

The SPKI authorisation certificates and their revocation methods have been developed by the Internet Engineering Task Force (IETF). The theory behind SPKI has reached the status of experimental RFC [Ellison & al, 1999], although the latest document has expired. The structure of the SPKI authorisation certificates has been derived from the theory document, but the document has not been completed and is therefore not an RFC. The revocation mechanisms discussed in this paper are based on the latest draft of the structure document, but we have also included the proposed changes from Kortesniemi, Hasu and Särs [2000].

An SPKI authorisation certificate is essentially a ticket granting the specified right to the indicated recipient. The certificate is always valid and can be used an unlimited number of times, unless its validity is somehow limited by listing conditions in the validity field of the certificate. Once the resource owner issues a certificate, there is no practical method of getting it back from, say, a misbehaving recipient, so the issuer needs to include some limiting conditions in the validity field when the certificate is created. And here lies the difficulty: all possible future problems have to be anticipated and suitable countermeasures must be devised at the creation time. To better appreciate the problems involved, let us have a look at the various revocation methods available. Although there are six different methods to limit the validity of a certificate, only four (types C, D and E) can be considered revocation methods; the rest are just validity management methods. In this paper, we have grouped the methods in five types based on the speediness of revocation and expiration characteristics (Table 1). We can say that the types refer to the validity mechanisms themselves, or to certificates, whose most effective mechanism is of the type mentioned.

Table 1: The SPKI validity management methods

Type	**Method**	**Speed of Revocation**	**Notes**
A	No `Validity Period` / Only beginning time (= no end time)	N/A	Does NOT expire
B	End time / complete `Validity Period`	N/A	
C	`Renew`	After current certificate expires	
	`CRL`	After current `CRL` expires	
	`Reval`	After current “Bill of Health” expires	
D	`One-time`	Immediately	Can limit the usage of a group of users
E	`Limit`	Immediately	Can limit the usage of the particular user

The simplest method is a `validity period`: the certificate is valid only between the dates stated. However, both of the dates (not-before and not-after) are optional, so it is possible to create e.g. “eternal” certificates by omitting the expiration date (type A). Such certificates should be used only with careful consideration and are not likely to be seen by end-users (example: a computer granting the administrator all the rights to the computer). Once the validity period also contains an end time, we have a more regular certificate (type B). These kinds of certificates are good when the value of the right or the risks from misuse are not significant (example: one-day bus ticket).

When the value or the risk is high, we need some way of revoking the certificate. First, we look at type C that has three methods. `Renew` divides the long validity period into several shorter ones that are represented by individual certificates, and provides an automated method for fetching the subsequent certificate after the current one has expired. The issuer can at any time stop the distribution of new certificates so after the current (short lived) certificate expires, the right is revoked. `CRL` (Certificate Revocation List) is based on a periodically published list of revoked certificates. Revocation takes effect as soon as the subsequent list has been published (i.e. after the current one expires). `Reval` is based on a periodically published "bill of health", which assures that a certificate is still valid. Without it, the certificate is invalid. Unlike `CRL`, `Reval` is not a list; it is issued individually to each certificate. These three methods appear similar to the issuer: the revocation takes place after a delay. This makes them suitable for a situation, where revocation is required, but the speed of revocation is not essential (example: one year bus ticket, which can be revoked every two weeks). The revocation methods of types D (one-time) and E (`Limit`) require contacting an online server every time the certificate is used. They can also be used to control the amount of use, not just to revoke the certificate completely. One-time can limit the usage on a general level (example: there are only 50 parking spaces in the garage, so limit the number of cars to 50), whereas `Limit` can actually control each individual certificate (example: this certificate allows 10 bus trips). The advantage of these methods is that revocation takes place immediately, but they also require a network connection.

4. Cases

In order to implement a system that is based on certificates, the user needs to have a terminal into which the certificates are loaded. Here, we consider a hand-held device type of terminal that has a screen and an ability to communicate with other systems either wireless or via a physical interface.

4.1 Bus Tickets

Helen, 15, wants to go shopping downtown, so she gets on a bus and buys a single ticket that is loaded onto her PDA. The ticket is valid for one hour, during which time she can transfer freely. In downtown, she needs to transfer to a tram to get to her favourite department store. This ticket scheme can be implemented with type B certificates. Helsinki City Transport (HCT) has delegated the right to sell tickets to all the buses and kiosks by issuing certificates to them. When Helen gets on the bus, the ticketing system of the bus creates a new certificate that is valid for one hour. The new certificate, together with the bus's certificate, is loaded onto Helen's PDA, which already contains a software module that can talk with the bus's system, and Helen's account is charged. When Helen transfers to the tram, the tram's system talks with Helen's PDA to check that she has the right to enter the tram. This ticketing scheme is rather simple. No revocation mechanism is needed, because the value of the ticket is very small. If Helen loses her PDA (and her ticket), the value of the ticket is her least concern. The issuer, HCT, does not have any interest in being able to cancel customer's tickets, because customers pay for their tickets in advance, and the customers cannot break any agreements that should result in the termination of the ticket.

Helen's father Matt wants to get to work in the morning and back to home in the evening, five days of a week, and an annual bus ticket is a good solution for him. He will not have to keep on buying new tickets, and an annual ticket will probably cost much less than, say, 500 single tickets. However, Matt is a forgetful person, and since an annual ticket costs around 600 €, he is worried about losing the PDA and the tickets with it. He needs some way to ensure that if the PDA does get lost, he will be able to revoke the ticket and get a refund for it. All this can be achieved with type B certificates, but for the user, losing the ticket involves too great a risk.

In order to provide good and safe service for its customers, HCT should be prepared to issue new tickets to replace lost or stolen ones, so HCT should use a method where tickets can be cancelled. This requires the use of type C or type D methods.

With the type C methods, tickets cannot be cancelled immediately but only after a period of waiting. The `CRL` method requires that the list of cancelled tickets is updated onto every bus every morning, i.e. it requires a periodic connection to a central server. As a result, it may take up to 24 hours before a ticket becomes invalid. This system works well when there is only one issuer (HCT). It also requires that tickets cannot be resold or delegated, since that would mean more issuers. If there are many issuers, it gets impractical or impossible to obtain all the possible `CRL`s in advance, or alternatively, the required `CRL`s must be obtained online at the time when the ticket is used. In the simple case, the end-user's terminal is not required to go online, and the bus's system needs to go online only occasionally. The next method, `Renew`, is based on dividing the annual ticket into, say, two-week periods. At the end of each period, the ticket is renewed until one year has passed. In this case, we don't need a list of all cancelled tickets on every bus. Instead we get the problem of renewing the ticket. The ticket must contain the address of an online server where the renewed ticket can be obtained. This kind of renewal must not be the user's responsibility, since it does not match the user's goals. Let's consider the simple case. If Matt gets on the bus and the ticket is in the middle of a two-week period, everything goes well. The ticketing system on the bus checks Matt's ticket and lets him in. If one two-week period is coming to an end, Matt's ticket needs to be renewed. This can be done automatically by the user's terminal. The user's terminal should contact the online server and ask for a new ticket. When Matt gets on the bus next morning, everything is fine again, and Matt does not need to know anything about the renewal process. Alternatively, if Matt's terminal is unable to go online, the bus's ticketing system can renew the ticket when Matt gets onto the bus.

The third method, `Reval`, is based on revalidating the certificate again and again. It works very similarly to `Renew`, but here, the user's terminal must supply the ticketing system both with the certificate that is valid for one year and with a "bill of health" that is valid for two weeks at a time. Instead of renewing the certificate itself, the bill of health must be renewed, which makes the system more complex. From Matt's point of view, all the three different type C mechanisms are the same, so they should also look the same. The differences are technical and understanding them does not provide any added value for the end-user.

The problem of type C certificates is that they cannot be revoked immediately. This is both a usability and a utility problem. The user may have difficulties in understanding or accepting that his electronic ticket cannot be cancelled immediately. With type D certificates, where the certificate requires that its validity be checked from an online server each time the ticket is used, the ticket can, instead, be cancelled immediately. However, using type D certificates requires that the resource is always online when it is used. Alternatively, the user's terminal could be required to go online, but this would be poor service and poor system design. When we look this from Matt's point of view, we see that for Matt, the only thing that matters is that he does not have to bear the risk of losing his investment in the yearly ticket. This can be achieved with type C certificates by making a business decision. HCT could issue a new ticket for Matt if the old one is lost or stolen. Possibly HCT could charge a small fee for this service. HCT could then itself bear the risk that somebody uses Matt's ticket before it can be revoked.

HCT could make this business decision, because this could be more economical than implementing a type D system, where all the buses must be online at all times. Also, the speed of the process is important for both Matt and HCT. If passengers must wait before they can enter the bus, it is irritating. If an actual implementation of type D system turned out to be too slow, a type C system should be used to ensure good quality service.

4.2 Parking House

Jane, Matt's wife, works for a company that rents 50 parking spaces for its employees from a nearby parking house, because it is known from experience that there are normally about 45 cars present at any one time, although there are almost 100 employees. The company then needs to distribute the parking permits to the employees. With the `One-time` method, we can implement a system for the garage that allows resource pooling and efficient use of parking lots. Every employee's PDA stores the certificate that grants access to the parking house. At the garage door, Jane's PDA automatically communicates with the door. The door lets Jane in only if less than 50 lots are in use. If all the rented spaces are occupied, the door tells Jane that all the spaces are full. For Jane, it is important to be able to know how many lots are still free. Therefore, the server that verifies the validity of the parking ticket (by checking the number of used lots) should also provide a facility to check the number of free lots. If Jane should decide to leave the company, her certificate can be immediately revoked, thus preventing any further use of the garage and without affecting any of the other employees.

4.3 Charge Card

Helen is about to leave for a two-month language course in London. During the course, Helen will need money for food, travelling, souvenirs and activities like theatre. Matt concludes that 300 € should suffice nicely and gives Helen access to his bank account for that amount. With a traditional charge card or a credit card, this would mean going to the bank and ordering a separate card for Helen (which can take several days). Further, such a card cannot be limited to any amount, so theoretically Helen could empty Matt's account.

When using certificates to implement a paying scheme, we can accomplish the limit [Heikkilä-Laukka]. Matt can go to his PC, open the bank program (used to pay bills) and create a "charge card" (certificate) for Helen, without any assistance or delay from the bank. These "charge cards" are stored in the user's PDAs and are used in shops to pay for products and services. For the certificate Matt needs Helen's "ID number" (public key), which is kept inside Helen's PDA. Helen gives the ID number by bringing the PDA next to the PC. Matt then keys in the limit (300 €), chooses the account from which the money should come, and finally sends the finished certificate to Helen's PDA. Technically, the limit of 300 € is implemented using the `Limit` method, which means that somewhere there is a server that monitors the usage of Helen's certificate. Theoretically, this could be any server, but in practice giving that choice to Matt would just make the system more difficult to use. Hence, it is quite natural that the server belongs to the bank, which Matt trusts already. The choice about which server to use must be made by the designer of the system, and Matt does not need to know the details.

During her course Helen can pay for her expenses by using the PDA at the cashier. The due amount appears on the PDA's screen, and once Helen accepts the sum, the money is transferred to the shop. Towards the end of Helen's course she finds out that a trip to Paris is being organised, but she no longer has enough money to participate. She decides to call her father and ask for a higher limit on her charge card. At that time, Matt is enjoying his summer vacation at the family cottage. Luckily, he has his PDA with him, so he can use it to raise the limit to £400 by simply typing a new value over the old one in the bank program. In Paris, the unexpected happens: Helen is pick-pocketed and she loses her PDA. She immediately calls her father, who can revoke the certificate at once.

In the case of a bank account/ charge card, only the `Limit` method can be used to control the amount of money spent on that certificate. If Matt had decided not to impose a limit, the bank's software would have chosen the `One-time` method instead, because the possibility of losing one's PDA still requires the possibility of immediate revocation. Implementing the system in this way is the responsibility of its designer. The system must not require that Matt make the decision about which kind of certificate to use. In order to make good decisions on the user's behalf, the designers must uncover the user's goals and needs before the system is implemented, and simulate their design from the end-user's point of view.

4.4 Summary of findings

Certificates can be used to enable various user tasks. In the cases just presented, the end-user does not need to be aware of certificates or revocation methods as such. They would be confusing by introducing new concepts that have nothing to do with the user's real goals. Instead, the end-user should be presented with concepts and information that match her goals: a bus ticket with an expiration time, a button to cancel someone's right to use one's bank account, or a phone number to call when the bus ticket or credit card is lost.

The designer of a system, however, has to understand certificates and revocation methods. It is the designer's responsibility to choose a suitable method with which to support the user's goals and to analyse which options are relevant to the end-user. To do this, the designer probably has to conduct field studies and observe and interview end-users. In particular, in each presented use case the required functionality could be reached with only one or two different revocation methods. Therefore, offering the end user a full list of methods is a bad idea. We also noted that `CRL`, `Reval` and `renew` can be made to look identical to the end-user (except in a limited set of cases), so the choice between them can be based on technical reasons.

The designer should follow established usability and interaction design guidelines in creating the system. Especially visibility of the user's data and the system's state are important. The end-user must be able to see what rights she has acquired, such as a bus ticket. The validity periods and limits for such rights should be shown. With the `Limit` method (type E), it is important that the server is able to tell the remaining limit, in addition to validation checking. The user must be able to easily see, to whom she has delegated rights, and revoked permissions must also be clearly visible. When presenting user's data and the system's state, all information should be shown in its context. The delegated right to use one's account must be presented together with the account in the bank application.

The two reasons for revocation – the issuer discovers that the receiver is misusing the certificate and wants to take it away, or the receiver loses the certificate and wants it replaced, in which case the old one has to be revoked – enable us to make the following conclusion on the different types. Type A certificates should not be offered as an option to end-users, because they are valid forever and cannot be revoked. Type B certificates are a good choice, when the value of the right is not significant and the end-user cannot cause significant damage to the issuer by misusing the certificate. Type C certificates could come into play when the value becomes so big that a revocation method is necessary, but the misuse does not require immediate revocation – for instance, when misuse is rare and the issuer can understand and bear the risk of misuse. This is how credit card companies handle misuse today. However, it must be noted that for the end-users it is important that they do not need to bear the risk. The system should be such that from their point of view, the revocation takes place immediately. Finally, the online methods should be used when immediate revocation is desirable. The choice then depends on whether we want to limit individual certificates (type E) or just the certificate group (type D).

An implementation issue with type C is the required online connection. If the validity information is fetched by the user's PDA, it should take place automatically so that the operation is transparent to the user. If the user has to take some steps to get the information, it makes the system appear much more complicated and means unnecessary work. Since currently many PDAs are not capable of online connections, it seems that the resource, e.g. the bus, should fetch the validity information. In this case, however, the resource has to be online all the time, because it usually cannot anticipate the required information, whereas the PDA only requires occasional connections. Only in limited situations where there are a very small number of `CRL` issuers (the HCT in the bus ticket example), can the `CRL`s be regularly fetched, and hence the connection is not required continuously. Another implementation issue is the feedback time. If the response time of a higher level revocation method (type D or E) is too long, say, over a couple of seconds in the bus example, the designer should choose a lower level method (type B or C), since immediate feedback and fast operation are important for the end-user.

5. Conclusions

On basis of the above analysis, we can clearly see that from the end-user's point of view, the revocation methods indeed have differences. For example, the types D (`One-time`) and E (`Limit`) can be recommended, since they provide the users with the greatest amount of control over the revocation management and are relatively easy to understand. However, we have also seen that the usability issues within revocation management are manifold and cannot be resolved without a thorough understanding of the specific use situations the system is designed for, nor without knowing who will be using these systems, where, when and how.

References

Adams, A, Sasse, M.A: Users Are not the Enemy. Communications of the ACM, December 1999, Vol. 42, No. 12, pp.41-46.

Ellison, C, Franz, B, Lampson, B, Rivest, R, Thomas, B and Ylönen, T. SPKI certificate theory. Request for Comments: 2693, September 1999.

Heikkilä, J, Laukka, M: SPKI based Solution to Anonymous Payment and Transaction Authorization, Proceedings of the Fourth Nordic Workshop on Secure IT Systems (Nordsec'99), 1999, Kista, Sweden

Kortesniemi, Y, Hasu, T and Särs, J: A Revocation, Validation and Authentication Protocol for SPKI Based Delegation Systems, Proceedings of Network and Distributed System Security Symposium (NDSS 2000), 2-4 February 2000, San Diego, California

Whitten, A. and Tygar J.D: Why Johnny Can't Encrypt: A Usability Evaluation of PGP 5.0. Proceedings of the 8th USENIX Security Symposium, August 1999.