Delegation Protocols for Electronic Commerce

Bruno Crispo
Cryptomathic
Corso Svizzera 185, 10100 Torino, Italy
crispo@cryptomathic.com

Abstract

Many commercial and financial activities in the real-life require reliable mechansisms to provide accountability for the transactions that has been executed. If electronic commerce aim to substitute or support similar activities in the electronic world, it has to provide the same degree of accountability. Despite this requirement is crucial, we observe that many existing security mechanisms and protocols are not designed by considering accountability as a fundamental property. In this paper, we show that this is true, particularly in the case of delegation protocols. Then to address the problem, we explicitly introduce accountability in delegation by defining a new semantics and by designing a new protocol that help to eliminate this lack of accountability.

1. Introduction

Electronic commerce over wide area networks has the potential to change the way many business are conducted in future. Thus, there is growing interest in developing applications for conducting commercial transactions. However, in most legal and commercial applications, the ability to hold individuals or organisations accountable for transactions is crucial. Consequently in order to be effective, electronic transactions must be able to provide the same degree of accountability as the real-life transactions that they intend to substitute. Despite the crucial role of accountability, many existing security primitives and protocols fail to unambiguosly prove the association between entities of the system and statements made by these entities.

Electronic commerce applications are usually distributed and the resources required to carry out an operation are rarely all local to the system in which the user is logged in. Consequently, delegating tasks and actions to other, physically remote, components of the system is more often the rule than the exception. Besides and not less important, the number of activities and accountability performed through the use of computer systems has increased to the extent that users often cannot personally perform an action but need to delegate some of their tasks to somebody else. These considerations motivate our analysis of the general problem of accountability in the particular context of delegation.

This paper is organised as follows. Section 2 introduces the notation used. Section 3 defines the basic notion of accountability, rights and security policy. Section 4 and 5 propose a new semantics of delegation and compare it with the existing semantics of delegation of rights. Section 6 discusses trust assumptions and specify a set of design priniciples, followed by the protocol specified in Section 7. Section 8 review the related work. Section 9 and 10, respectively propose future work and conclude the paper.

2. Notation

We use the term principal to denote an entity of the system. They are denoted with capital letters A, B, etc. The principal that delegates is said to be the *grantor*. The principal that receives delegation is said to be the *grantee*. The target that accepts the last operation invocation of a chain of delegations is called the *end-point*.

In this paper we will use only public-key cryptography and we will use *SK* to denote a signing key and *VK* to denote a validation key. The owner of the key is denoted by a subscript of her initial (i.e., SK_A denotes A's signing key). VK_A denotes A's validation key. We associate to each key the power [1] for which the key may be used and for which it is valid. Thus $SK_{A,x}$ denotes the signing key of principal A that is valid to exercise power x. We will explain in more detail this requirement and its motivation in Section 5.2.

The tuple *(SK_A,VK_A)* denotes A's key pair for authentication. The function *SK(M)* denotes the digital signature of message M computed by using the signing key SK. The function generates a bit string that represents the digital signature of the message given in input.

[1] We here use the term *power* as defined by Makinson [7] and Babak and Sergot [2] that is the ability to perform an operation. Note that the notion of power does not entiles the *permission* to perform that operation.

3. Accountability

We focus our attention upon applications and systems where accountability and auditing are always required. Our analysis and study of delegation is mainly driven by those two important properties.

We start by giving the following definition of accountability (Kailar [5]):

> **Defn.**
> *Accountability is the property whereby the association of a principal with an object/action/right can be proved, with very high probability, to a third party (i.e., a party who is different from the principals participating in the transaction that involved the object/action/right in question).*

We do not discuss here the details of the proof but we assume that such a proof can be somehow provided. Also, we do not use the definition of accountability in a legal sense. We cannot merely by technical means, establish who is going to be legally, morally or socially liable for an action. Our goal, more modest but still quite difficult, is to determine which principal is the most likely to have technically performed a particular action while exercising a delegated right. An unbiased observer, looking at the audit files that record all the actions carried out in the system, must be able to decide whether a particular action was executed (with very high probability of being right) by a specific principal and not by any other principal of the system.

4. Delegation of Accountability

In security literature (e.g., Gasser and McDermott [4], Kerberos [8], SPX [12], and Varadharajan *et al.* [13]), delegation, is almost always, merely defined as *delegation of rights*. This refers to:

> **Defn.**
> *the process whereby a principal A, authorises another principal B, to act on her behalf, by* ***sharing*** *a set of her rights with B, possibly for a specific period of time.*

Delegation of rights assumes that a principal A has herself a set of rights $\mathcal{R}$, and it delegates all or a subset of them, $\mathcal{R}'$, to another principal B who can then act, instead of A, to exercise that particular set of rights $\mathcal{R}'$.

There are many situations where it is necessary not only to delegate rights but also to delegate the accountabilities that may be associated with these rights. For those applications, we propose a new semantics of delegation that we call *delegation of accountability*, defined as follows:

> **Defn.**
> *delegation of accountability is used to refer the process whereby a principal A, authorises another principal B, to act on her behalf, by A* ***transferring*** *a subset of her security policy to B, possibly for a specific period of time.*

A transfer of security policy and, we will argue possibly but not necessarily, of rights occurs. We consider this issue very important because policies (how to use) jointly with rights (what to use) classify the various semantics of delegation. A portion of the grantor's security policy is transferred to the grantee. Usually, just a portion and not all the security policy is transferred because the grantor wants to retain at least some particular rights by herself, such as the right to revoke a previous delegation. We need to explain in great detail what we mean here with the word "transfer" [2]. The grantor executes this transfer by specifying the grantee as the principal entitled to exercise the set of rights during delegation. By *not* specifying herself, the grantor transfers to the grantee the semantics that apply to those rights within the boundaries of that specific instance of delegation. The transfer is necessary and sufficient to convey also the accountability for the rights that has been delegated. The grantor in fact, most of the time after delegation, still retain the practical ability to perform these rights, but not anymore with the semantics specified within that instance of delegation.

5 Design Issues

We have defined the new semantics of delegation of accountability. In the rest of the paper, we describe how to design a delegation protocol that implement such semantics. We start by analysing in great detail the design issues that underlie the implementation and we will specify a set of design principles that need to be satisfied to be able to implement accountability in delegation protocols. After this analysis we specify and describe the protocol itself.

5.1 Delegation Credentials

Basic to the mechanisms that implement delegation are the credentials used by the grantor to pass rights/accountabilities to the grantee. Principal A, to delegate to principal B, has to provide the latter with some credentials stating the authentication of A and the set of

[2] In this paper we refer to the Merriam-Webster's dictionary definition of these two terms. Thus share means to have, get, or use in common with another or others. Share usually implies that one, as the original holder, grants to another the partial use, enjoyment, or possession of a thing. Transfer means to convey from one person, place, or situation to another; to cause to pass from one to another; to make over the possession or control of.

delegated rights/accountabilities with the policy attached to them (i.e., validity period, etc.). The credentials we use, derive from the notion of *delegation token*, first introduced by Sollins [11]. A delegation token specifies the token's generator (the grantor), the grantee and the set of rights/accountabilities associated with the token and passed by the grantor to the grantee. A significant feature is that the verification code is produced by the grantor by signing the token with his signing key and producing a digest of the delegation token dependent on the data in the token. The token and its verification code constitute the credential that the grantor needs to pass to the grantee in order to delegate. The integrity of the delegation token, and its secrecy when needed, are enforced by using cryptographic tools. Furthermore, delegation tokens allow the delegation of rights/accountabilities among principals without the intervention of any server, because any principal can have the ability to verify the credential by himself, thus facilitating the provision of end-to-end mechanisms.

Delegation can be identity-based or key-based. With identity-based delegation, the token is passed to the grantee identified in the token itself, by her name or by her authentication key. Which key (if any) is then used for exercising the delegated rights/accountability is not specified. With key-based delegation (Gasser *et al.* [4]), the token is passed to a specific key possessed by the grantee, used for the sole purpose of exercise delegation. This specific key is called a *delegation key*.

5.2 Design principles

We introduce a set of principles that we will follow in our design. The first three principles apply to any protocol that implement delegation and they are *independence*, *principle of consent* and *robustness*. The other two *fine grained auditing* and *privacy of signing keys*, are instead, peculiar to the semantics of delegation of accountability introduced here.

> **Independence**
> *unnecessary dependencies among different security mechanisms must be avoided, in the design phase.*

In particular existing delegation protocols are weaken by creating unnecessary dependencies between authentication and delegation at the design phase. Even if we do not object to this, in principle (there may be specific, even if unusual, application where this is required), we want to point out that particular care must be taken in designing such protocols. A possible mistake (e.g., Low *et al.* [6]) is to use in the delegation token the same key that the grantee uses to authenticate herself also as delegation key. This may look very convenient because principals can be authenticated and delegate/been delegated by performing a single validation operation. However, this choice has a lot of drawbacks. It increases a lot the complexity of the key management and it weaken the robustness of the protocol. Concerning the key management issues, having two different key, one for the sole purpose of authentication and a separate one as delegation key, allows the validity period of authentication and of delegation to be independent. Also, this separation allows an easier implementation of role-based models. For example, in cases where several different roles has been delegated to the same grantee. The grantee may need more than a single delegation key, one for each role, while she has only one single key pair for the purpose of authentication. Besides at the time at which the authentication key is bound to a principal it is usually unknown whether delegation will be required, and which rights/accountability will be delegated, thus inconsistencies may arise. For instance, if the delegated rights/accountability have a different lifetime compared to the authentication key, the renewal of the key will be then very cumbersome.

So it is a good practice to design authentication and delegation as separate mechanisms.

> **Principle of consent**
> *nothing can be delegated if not explicitly accepted by the grantee.*

This important principle has been originated by Abadi *et al.* [1] and derives from the *least-privileges principle*. Delegation protocols first must require the grantor to specify which rights/accountability of hers she wishes to delegate to her grantee *and* second must require the grantee to explicitly accept these rights/accountability. This requirement reduces sensibly the possible abuses of the power of delegation of the grantor and prevents hidden transfer or sharing of rights/accountability. The grantee must be always aware of what she can be held accountable for. Without implementing this principle, the grantee has to blindly trust all the principals that have capabilities to delegate something to her not to misuse those capabilities. The principle of consent greatly reduces the extent to which the grantee has to trust all the possible grantors. This principle is a necessary but not sufficient condition to implement accountability. As we said, it stops some possible abuses from the grantor. However, once the grantee has explicitly accepted delegation, this principle does not prevent the sharing of rights and accountability between grantor and grantee. After the grantee has accepted delegation, the (mis)trusted grantor may have the possibility to masquerade as her grantee for that instance of delegation, because they both may have knowledge of the delegation key used to exercise the delegated rights for that specific instance of delegation, so he can always forge delegation tokens. The grantor could forge a request of a service to the end-point, that appears as if it was made by

the grantee. On the other hand the grantee may have the possibility to repudiate transactions made acting as grantee, blaming the grantor for them, just by exploiting the above argument to defend his claim.

In electronic commerce applications, robustness against possible attacks and/or accidents, and confinement of trust and thus confinement of the extent of a failure/attack, are important goals to achieve whenever it is possible. Thus we apply the following principle:

> **Robustness**
> *secure protocols must designed to prevent attacks and/or failure or at least such that the extent of damage caused by successful attacks or failures is minimised.*

Delegation keys are particularly important because they help to prevent and to confine the possible damage of the following attack. If delegation tokens specify only the grantee's identity but do not use delegation keys, an attacker that succeeds in masquerading as the grantee (i.e., who knows the signing key the grantee uses for authentication) he can automatically exercise all the rights that have been delegated to that grantee. Whereas if delegation tokens specify also the delegation keys the attacker, to be successful, has to masquerade as the grantee *and* be able to use the delegation key at the same time. If successful in both his attempts, he can abuse however, only the rights bound to that particular delegation key. This of course, assuming that a different delegation key is used for each instance of delegation.

Our goal is not only to confine damage but, in case of delegation of accountability, we should be able to trace it. To achieve that we follow the principle:

> **Fine grained auditing**
> *finer the granularity of the actions performed by the system, the higher the possibility to detect those principal accountable for such actions.*

Accordingly with that, a different key should be associated to each different type of power in order to facilitate tracing of accountabilities. We denote a power as a couple [*operation(attributes),target*] where *operation* can have attributes associated with (e.g., [read(not forwardable),/usr/local/pub]). This couple specifies the power to perform the *operation* over the *target*. We distinguish very carefully all the powers that are defined and involved in the delegation process. We define separate powers to confer other powers, to accept them, to exercise and to revoke them. For example if P is a power, then the power to delegate P is another power: P', and the power to accept P is another power: P''. We consider this practice very useful, especially in prototyping security protocols, because it helps to clarify the distribution of trust and the extent of damage caused by possible attacks and failures (i.e., corruption of a cryptographic key).

The last principle is:

> **Privacy of signing keys**
> *the knowledge of a signing key is never shared between principals.*

Delegation keys and how they are handled are crucial for delegation of accountability. The way in which most of the existing mechanisms (e.g., Kerberos [8], DCE [9], SPX [12], the schemes proposed by Sollins [11], by Gasser *et al.* [4], by Varadharajan *et al.* [13], by Ding *et al.* [3]) manage these keys is one of the main reasons why they cannot implement delegation of accountability. All these mechanisms assume that the delegation key is chosen by the grantor and then somehow shared, preserving its secrecy, with the grantee. The grantee has to trust the grantor not to abuse her knowledge of the delegation key. If delegation of rights is being implemented this sharing does not cause any problem. However, in case of delegation of accountability the situation is different, because the sharing of delegation keys between grantor and grantee leaves doubts about who could have exercised the delegated rights by using the delegation key. This clearly introduces the opportunity to deny actions performed by using delegation. The well-known problem of repudiation is transferred to delegation. By sharing the knowledge of the delegation key, grantee and grantor also share delegated rights and their accountability. We design our protocol in such a way that always, the grantee chooses her delegation key pair and she never shares the signing key with other principals.

6. Delegation of Accountability Protocol

The protocol we propose is based on tokens of delegation (Gasser *et al.* [4], Sollins [11], Low *et al.* [6]). Our protocol allows principals to delegate their own accountability to any other principals. It assumes that each principal can generate public-key pairs and access to a digital signature service. The signing key is kept secret and/or its use restricted while the validation key is public. Moreover each principal can get the verification key(s) needed to verify digital signatures that she may receives, this include also keys used for authentication purposes. We do not specify in our description the part concerned with authentication of principals, which we assume already done when the delegation protocol starts. The delegation protocol is specified as follows:

1. G $\rightarrow$ g: G,g,M, SK_G(M)
where M=[G wishes to delegate to g accountability for Ω using $VK_{G;P\_t\_D}$]

2. g $\rightarrow$ G: g,G,M', SK_g(M')
M'=[g will accepts Ω using $VK_{g;P\_t\_A}$ and exercise Ω using $VK_{g;P\_t\_E}$]

3. G $\rightarrow$ g: T = G,g,M'', $SK_{G;P\_t\_D}$(M'')
M''=[g, G, Ω, $VK_{G;P\_t\_D}$, $VK_{g;P\_t\_E}$, $VK_{g;P\_t\_A}$]

4. g then signs T producing the delegation token:
{T, $SK_{g;P\_t\_A}$(T)}

where G is the grantor, g is the grantee, Ω = ($\mathcal{P}$,Δ), $\mathcal{P}$=set of tasks or roles and Δ=life span of the delegation token, P_t_D stands for "power to delegate", P_t_A stands for "power to accept" and P_t_E stands for "power to exercise". SK_G and SK_g are respectively the authentication key of grantor and grantee, ($SK_{G;P\_t\_D}$, $VK_{G;P\_t\_D}$) is the key pair that allows the grantor to delegate accountabilities, ($SK_{g;P\_t\_E}$, $VK_{g;P\_t\_E}$) is the key pair (delegation key pair) used by the grantee to exercise them, ($SK_{g;P\_t\_A}$, $VK_{g;P\_t\_A}$) is the key pair she uses to accept them. In message (3), a key rather than a name is used to identify the grantor so if an attacker succeeds to masquerade as the grantor he cannot fraudulently delegate grantor's accountability because he still does not know the key $SK_{G;P\_t\_D}$ necessary to be able to do it.

The grantor is the only one that can enable the grantee to use Ω, because the grantor has to sign the delegation token. When the grantee wishes to use the delegated accountabilities she must present, to the end-point, the delegation token

T, $SK_{g;P\_t\_A}$(T) = [M'', $SK_{G;P\_t\_D}$(M'')],
$SK_{g;P\_t\_A}$(M'', $SK_{G;P\_t\_D}$(M''))

followed by the request of the specific service she wants. The end-point will check the privileges carried in the delegation token against her access control policy. The end-point can be any principal of the system because the token is verifiable by all the components of the system. We have assumed, in fact, that an authentication service is available. Thus all the principals can verify the correctness of the delegation token after they get the grantor's and grantee's validation key from the authentication service in order to authenticate them in the first two messages of the protocol.

Our protocol satisfies the principles we introduced.

- **Independence**. We keep the key pair used for authentication ((SK_G,VK_G) and (SK_g,VK_g)) separated from the delegation key pairs used to exercise some power and being invested, consequently, by some accountability (e.g., ($SK_{g;P\_t\_E}$, $VK_{g;P\_t\_E}$)). Each principal may act in many roles or exercise many different tasks at the same time, thus she can have several key pairs of this latter type.
- **Principle of consent**. Message (2) and (4), of our protocol have the explicit purpose to implement consent of the grantee for the task Ω. Message (2) is used by **g** to express to **G** is availability to accept Ω, and to submit the set of keys **g** will use. In message (4) this acceptance is sealed.
- **Robustness**. By using delegation keys in the tokens that are different from the keys used for authentication (i.e., (SK_G,VK_G), (SK_g,VK_g), we do not only make the two mechansims independent, but we also enhance the robustness of the whole protocol as said in section 5.2.
- **Fine grained auditing**. We have defined a different key for each power ($SK_{g;P\_t\_D}$ to delegate, $SK_{g;P\_t\_A}$ to accept, $SK_{g;P\_t\_E}$ to exercise) involved in our protocol. Also, having distinguished between the key to accept accountabilities and the key to exercise them we provide a mechanism more flexible than the existing ones. The two keys do not necessarily have to be different but there may be cases where the principal that accepts the accountabilities could be different from the principal that will exercise them, thus our mechanism supports this occurrence without requiring secret keys to be shared.
- **Privacy of Signing Keys**. In contrast with other schemes, our protocols does not assume that the grantor knows the delegation key, $SK_{g;P\_t\_E}$, that is used to exercise the delegated accountabilities. Only the knowledge of the validation key $VK_{g;P\_t\_E}$, is shared. The grantee chooses this signing key and she is the only entity that knows it, providing in such a way a strong mechanism for non repudiation purposes.

Our protocol can be easily extended to the case of chains of delegation, preserving the same properties. For problem of space, we leave to the reader, the detail of such extension.

6.1. End-Point

AIn order to trace accountability is necessary to trace the dissemination of delegation tokens. However, it is equally important to trace where these tokens are used. Delegation tokens entails grantees with a power but not with the permission to exercise that power. Such permission is granted when the token, following a service request, is presented to the end-point and successfully checked against the access control policy in force. Only after this check, the delegation process is completed. Trace of these checks need to be stored as evidence to support accountability in the system.

6.2. Auditing Mechanisms

Because we delegate accountability, it is reasonable to assume that there are auditing mechanisms bound to the delegation protocol. It is in fact, of little use to talk about accountability if the system does not provide any record of what happened. This may not be necessary for preventing attacks but for providing the evidence subject to investigation during a possible dispute. The existence of the transaction itself needs to be proved before it can be disputed.

It is worth to point out that, even in the case of absence of auditing mechanisms, our protocol still succeeds to deliver important guarantees. It will be in fact, anyway impossible, (due to our protocol and not to auditing), to successfully forge a grantee's request that implies delegated accountability, and this because, the grantee is always the only entity of the system that possesses the capabilities (i.e., $SK_{g;P\_t\_E}$) to generate such a request.

6.3. Recovery Measures

In case the run of the protocol is not completed, because of accidental failure or of denial of service attack, care must be taken about how the system will recover from this situation. If the protocol fails at step (1) or at step (2), grantor or grantee will discover it, because they do not receive the expected reply within a prefixed time period. In this case, they revoke the keys they generate (grantor $VK_{G;P\_t\_D}$, grantee $VK_{g;P\_t\_A}$ and $VK_{g;P\_t\_E}$), or they may simply rely on the natural expiration of Ω. The revocation is necessary to avoid the possible situation of having different genuine messages that grant the same powers to different keys, in such a way making auditing more difficult. If the protocol fails at step (3), the grantor has no way to directly recognise this situation because the grantee never replies to the grantee. There is no general fix to this problem, which we believe must be tackled on per-application basis. One solution could be to require the grantee to acknowledge, at step (3), the grantor. This solves the problem only partially, because the failure may now happen during the exchange of this acknowledgement, in which case the cited timeout solution and revocation of all the keys previously sent must be applied here as well. Another possibility is for the grantor to check via the end-point, where the grantee is supposed to exercise delegation. The presence of a correct service request, sent by the grantee to this end-point, implicitly guarantees that step (3) was successful. In step (3), the grantor, before signing the delegation token, may be required to check if $VK_{g;P\_t\_A}$ and $VK_{g;P\_t\_E}$ have been revoked, The same applies for the grantee. In step (4) the grantee may be required to check the validity of $VK_{G;P\_t\_D}$ before accepting the token. Again, these checks can be avoided by relying on the natural expiration of Ω, if specified.

7. Future Work

In this paper, we have defined informally, a new semantics of delegation. We chose an informal treatment because it helps to present a new concept in a more intuitive and easy to understand manner. However, as future approach we are interested also to rigorously formalise these semantics and reasoning about them.

Similarly, also a rigorous analysis for the protocol we proposed, is required. We will consider the use of both logic (e.g., Kailar [5]) and the inductive approach (e.g., Paulson [10]), to analyse our protocol and thus providing some form of assurance.

Finally, we are actually working to integrate our delegation protocol in the Java JDK 1.2 by using key-based certificates as delegation tokens.

8. Conclusions

In this paper we have drawn the attention to the broad issue of accountability and its impact on the design of secure systems. In particular, we considered the problem of delegation and shown that many existing semantics were defined and delegation protocols designed without considering the property of accountability. We introduced a new semantics that we called delegation of accountability and we have shown what are the differences between this new semantics and the semantics of delegation of rights. The new semantics is particular useful for electronic commerce applications, where a limited amount of trust between the principals of the system can be assumed. We then defined a set of principles (independence, principle of consent, robustness, fine grained auditing and privacy of signing keys) that need to be observed to be able to design mechanisms capable of implementing such a new semantics. We finally provide a practical example of such a design.

References

[1] M. Abadi, M. Burrows, B. Lampson, and G.Plotkin. A calculus for access control in distibuted systems. *ACM Transaction on Programming Languages and Systems*, 15:706–734, September 1993.

[2] S. Babak and M. Sergot. Power and permission in security systems. In *Proceedings of Security Protocols International Workshop*, volume 1796 of *LNCS*. Springer-Verlag, 1999.

[3] Y. Ding, P. Horster, and H. Petersen. A new approach for delegation using hierarchical delegation token. In *Proceedings of the 2nd Conference on Computer and Communications Security*, 1996.

[4] M. Gasser and E. McDermott. An architecture for practical delegation in a distributed system. In *Proceedings of the IEEE Symposium on Security and Privacy*, 1990.

[5] R. Kailar. Reasoning about accountability in protocol for electronic commerce. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 236–250, 1995.

[6] M. Low and B. Christianson. Self authenticating proxies. *Computer Journal*, 37:422–428, October 1994.

[7] D. Makinson. On the formal representation of right relations. *Journal of Philosophical Logic*, 15:403–445, 1986.

[8] B. Neuman. Proxy-based authorization and accounting for distributed system. In *Proceedings of the 13th International Conference on Distributed Systems*, May 1993.

[9] J. Pato. Extending the dce authorization model to support practical delegation. Technical report, Open System Group, June 1992. OSF DCE RFC 3.0.

[10] L. C. Paulson. The inductive approach to verifying cryptographic protocols. *Journal of Computer Security*, 6:85–128, 1998.

[11] K. Sollins. Cascaded authentication. In *Proceedings of the IEEE Conference on Security and Privacy*, April 1988.

[12] J. Tardo and K. Alagappan. Spx: Global authentication using public key certificates. In *Proceedings of the IEEE Symposium on Security and Privacy*, 1991.

[13] V.Varadharajan, P. Allen, and S. Black. An analysis of the proxy problem in distributed system. In *Proceedings of the IEEE Symposium on Security and Privacy*, 1991.