DIVISION OF COMPUTER SCIENCE

Discretionary Non-Repudiation

Marie Rose Low

Technical Report No. 267

September 1996

Discretionary Non-Repudiation

Marie Rose Low, James A. Malcolm

University of Hertfordshire

September 1996

Abstract

Non-repudiation is necessary when services are provided in a distributed and open computer system. In this paper we aim to clarify what is meant by non-repudiation, identify who needs it, and determine why it is needed. Non-repudiation can be provided by having a globally available and trusted third party which will vouch for pertinent actions. This solution is not ideal in a global environment. We discuss the requirements for non-repudiation, and show that different solutions are necessary in different situations. We propose the use of Self-Authenticating Proxies (SAProxies) as a tool which would enable many of these solutions to be implemented.

1. Introduction

The term non-repudiation is used to mean that information provided by a principal in an exchange of data cannot subsequently be denied. In other words the data is attributable to the principal who originated it. For example if a statement, such as a request for a service was made, the request cannot later be denied.

This is not the only aspect of non-repudiation however. Within this notion is included the requirement that an exchange also needs to be proved to have occurred. Thus if a statement has been made by one principal to another the receiver of the statement cannot deny that he heard the statement e.g. when a request is made for a service the service provider must not be able to deny receiving the request. It is important to note that providing non-repudiation does not necessarily prevent malicious behaviour, but instead, it serves to prove the identity of the offender.

In the next section we discuss the need for non-repudiation. This is followed by a description of a non-repudiation scheme proposed by Coffey and Saidha in [2]. In section 4 we describe an alternative scheme using Self-Authenticating Proxies (SAProxies) [4] [5]. Section 5 is used to illustrate how SAProxies can be applied to provide non-repudiation. Section 6 is a summary of issues discussed and some of the conclusions that can be drawn from them.

2. Who Needs Non-Repudiation?

We wish to consider issues concerned with non-repudiation in a wide context in a distributed environment. In order to generalise our discussion we employ the phrase a 'principal makes a statement'. When using this phrase, we include situations such as where a principal 'writes' (is the author of) data, requests and provides a service or resource, pays for a service or resource or simply makes a statement.

Non-repudiation is necessary both to attribute a statement to an author and to prove that the statement is received when sent. Proof of origin of an electronic statement is typically provided by the use of digital signatures [3]. Digital signatures bind the identity of the author of the data with the data in such a way that the binding is unforgeable and undeniable and the data is

tamper-proof. We consider the need for non-repudiation in areas where it is of great importance, as in authorisation schemes where only those in authority can grant access, in access control schemes where access to protected resources must only be allowed to particular principals, as well as in resource funding where payment for resources is required. In all these cases it is necessary to be able to attribute actions to their perpetrators so that genuine statements can be recognised and impostors cannot behave maliciously without detection. The reasons for requiring non-repudiation may influence the way it is provided. These requirements are varied and a flexible scheme is necessary to cater for the needs of parties wanting to carry out business transactions across a distributed network, each working within the control of their own domain.

Having illustrated the double nature of non-repudiation, it is important to note that proof of origin and proof of receipt are not both necessary in all situations. For example, if a statement is the provision of a service then the receiver may not be interested in where he gets the service from. However, the provider of the service is probably very keen to get proof of receipt so that he can claim payment for the service he provided. In this case it is possible that the receiver may deny receiving the service, thereby preventing the provider of the service from claiming payment.

On the other hand, literature or software may be freely available on the Internet. In this case the provider of the information is not interested in who reads or downloads the information, but the receiver will very likely need to have proof of origin to ensure that he is getting what he is asking for and not information which is bogus or has been tampered with. The notion of proof of origin is necessary when it is important to know who made a statement and does not need to be tied to proof of receipt.

3. Non-Repudiation with Mandatory proof of Receipt

Let us now consider a scheme proposed by Coffey and Saidha in [2]. This scheme deals with the exchange of data between two principals and addresses the need to prove that an exchange has actually occurred. Two requirements are identified, proof of origin (POO) of the data and proof of receipt (POR) by the recipient. Attribution of data to a principal, POO, is achieved by using digital signatures [3]. POR is achieved with the use of two third parties as well as digital signatures.

The basic principles behind this scheme are as follows. Two entities, a Non-Repudiation Server (NRS) and a Time-Stamping Authority (TSA), which are deemed to be trustworthy and therefore trusted by both principals, are used to enforce the mandatory POR. The TSA provides a time-stamp to show the time at which a message is signed. The NRS acts as an intermediary between the two principals, so that the two principals do not exchange data directly, but through the NRS. The NRS receives the data with the signature and TSA time-stamp, the POO, from the sender and negotiates a POR with the receiver. When the NRS holds a valid POO and POR it then sends the POO, which includes the data, to the receiver and the POR to the sender.

A protocol for the exchange of messages between the parties is described and simple rules can be applied in an arbitration procedure to settle disputes.

The benefits of this scheme have to be assessed against the requirements for non-repudiation. Whereas the scheme described above is perfectly adequate for an exchange of data in a centralised environment where both POO and POR are necessary, this scheme cannot readily be adapted to enforce non-repudiation when a service or resource is being provided as opposed to an exchange of data. A NRS can neither withhold a service until payment is guaranteed nor ensure that it will be provided after it has been paid for.

The scheme necessarily provides both POO and POR and may limit the use of the scheme to applications that require both POO and POR. It is too rigid to be applied to situations such as those described in the previous section. There are also other features which need to be considered if this scheme is to be employed. As the authors point out, in a distributed system there is a need for a globally trusted NRS, or a hierarchy of trusted NRSs. Thus,

communicating parties must have a trusted link between them and without such a link non-reputation becomes impossible. A hierarchy of NRSs becomes essential in a distributed environment. Establishing a global service is prohibitive and inhibits the use of the scheme for business transactions between principals who have no previous knowledge of each other such as on the Internet. The users of the NRS service also become vulnerable if the NRS misbehaves or if it is attacked by any party. Furthermore, this requires on-line communications between the parties and the NRS which may lead to denial of service if there is a communications failure. The NRS also becomes a bottle-neck if its services are over-subscribed.

It is clear that when non-repudiation is required in a general context, then a more flexible means of achieving it must be provided. We suggest that the contractual agreement between any two principal is best worked out and enforced by the participating parties rather than by a third party. In the following section we address the problem of non-repudiation in the context of granting access to, providing and paying for information or resources. Within this context, the straightforward exchange of data is obviously included. We now describe an alternative scheme which may be applied to enforce non-repudiation in a general situation.

4. The SAProxy Scheme

Let us introduce the SAProxy authorisation scheme [4] [5] and consider it as a means of providing non-repudiation. The SAProxy scheme is a mechanism based on tokens of authorisation and public key encryption. An SAProxy token is an extended capability naming the principal to whom it is issued, the object in question and the access rights and is digitally signed by the issuer of the SAProxy.

This scheme allows principals to define and delegate access rights, for objects over which they have authority, to a particular party. This is done in such a way that all authority granted to access information, a service or resource, is attributable to the party that granted it. All requests for access using SAProxies is also attributable to the requester.

SAProxies may be bound together in order to express further delegation of authority and combined authorisation. Thus, a SAProxy field may refer to another SAProxy instead of holding a real value. SAProxies are referred to by their digital signatures which provide a unique and verifiable pointer mechanism.

The authority, expressed in SAProxy tokens is unforgeable as the tokens are digitally signed and can be produced and verified locally by each principal. There is therefore no need for a globally trusted central authority. Only the principal to whom the token is issued can use it, so there is no problem with propagation. Requests together with the authorising SAProxies can be recorded to provide an audit trail of the actions of all principals. These features make SAProxies particularly suitable for a distributed open environment. The following is a brief description of the mechanism. For a full description see [6].

4.1 The SAProxy Mechanism

SAProxies are based on public key encryption, so all principals involved must have a public key pair. Public keys are certified and signed by certification authorities.

A principal can then be identified and referred to by his public key certificate (PKC), because these are unique, and the PKC itself can be referred to by its signature block, e.g. a user A can be referred to by his PKC signature, $sigC_A$ [1].

SAProxies, represent the access rights to an object as delegated by the issuer of the SAProxy, to a party named in the SAProxy. Each SAProxy is signed by the issuer and the signature block forms part of the token. SAProxies are also referred to by their signature block. SAProxies have the following fields: the identity of the principal it is issued to, the identity of the issuer, the object (resource/ information), the access rights that are being granted to that object and the signature block.

To illustrate the use of SAProxies, consider a principal A, with PKC C_A, which has authority over an object and wishes to grant write and read access to principal B with PKC C_B. The SAProxy, $D_{B:A}$, would be as follows:

$D_{B:A}$: $sigC_B$ $sigC_A$ object *write/ read* $sigD_{B:A}$

where the SAProxy signature block is

$sigD_{B:A}$: {$sigC_B$ $sigC_A$ *object* *write/ read*}K_A^-

Anyone with A's PKC can use A's public key to verify that A signed the SAProxy and that it has not been tampered with and this can be done in a domain that is remote from the one in which the SAProxy was generated.

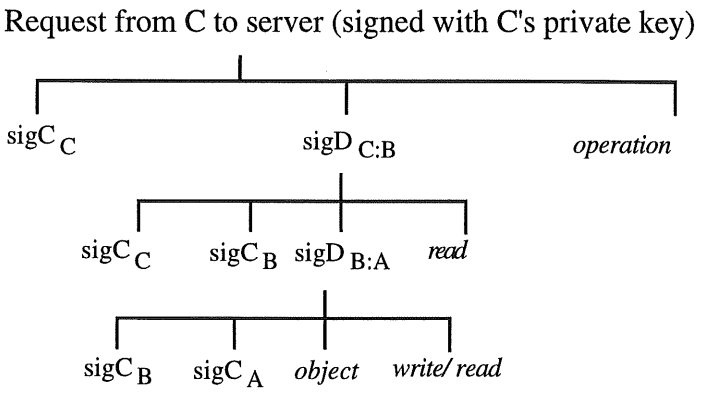
Within a single SAProxy, the information shown above need not be explicitly stated but may be referred to by other SAProxies. If B delegates his own authority to C, B does not refer to the object explicitly, but refers to $D_{B:A}$. In doing so, B is automatically binding into his delegation of authority, proof of his own authority. This is shown in B's token, $D_{C:B}$, generated for C:

$D_{C:B}$: $sigC_C$ $sigC_B$ $sigD_{B:A}$ *read* $sigD_{C:B}$

The level of delegation allowed can be specified by having a SAProxy constraint in the access rights field which indicates how many times a SAProxy can be delegated. Thus these tokens of delegation can be bound together to show the grounds upon which access to an object is granted. It is important to note that because these SAProxies are signed by each issuer, they can be verified by any party, with the appropriate PKC, and their origin determined. Thus authority granted in a SAProxy cannot be repudiated by the grantor.

To read the object, C then sends his tokens of authorisation to the object server with his request. C has to send his SAProxy from B, $D_{C:B}$, and the authorisation SAProxy for B from A, $D_{B:A}$, together with the operation he wants to perform. He binds these together by signing them with his private key to show that he is the true issuer of the tokens.

The object server checks that the line of authority can be can be followed, that the access rights delegated are not enlarged and that the operation requested falls within those allowed to the requester (in this case only read is allowed). This is shown in the following tree of tokens:



All SAProxies are attributable to their issuers. A request from C to the server can also be formatted in terms of a SAProxy. C sends a request SAProxy, $R_{S:C}$, to the server S which has PKC, C_S.

$R_{S:C}$: $sigC_S$ $sigC_C$ $sigD_{C:B}$ *operation* $sigR_{S:C}$

This can be viewed as C delegating his authority to S to operate on the object on his behalf. This request is also attributable to its issuer. Thus, provided that the service provider keeps an audit trail of all requests and their authorising tokens then the POO of all actions (and not just data exchanged) is available. A principal cannot deny that he has made a request or that he has delegated authority which is what we are trying to achieve.

In the following section we will illustrate how to provide proof of receipt in terms of paying for a service received. We now show that the SAProxy scheme can be used as a payment scheme. Consider SAProxies that have in the access rights field the 'amount of funds' that the SAProxy owner is authorised to use. This SAProxy is like a physical cheque and can be used in much the same way that we use cheques and bank cards. The use of such SAProxies is dependent on the policies followed by the service provider and its clients. A cheque SAProxy, $Q_{S:C}$, would have the following information.

$Q_{S:C}$: $sigC_S$ $sigC_C$ *account* *amount* $sigQ_{S:C}$

C's bank, K, may endorse this SAProxy with another SAProxy, $Q_{C:K}$, in order to guarantee payment to the service provider.

$Q_{C:K}$: $sigC_C$ $sigC_K$ $Q_{S:C}$ *null* $sigQ_{C:K}$

This guaranteed cheque may be sent as payment to a service provider when a transaction has taken place or, as will be shown later, may also be bound in with the request SAProxy.

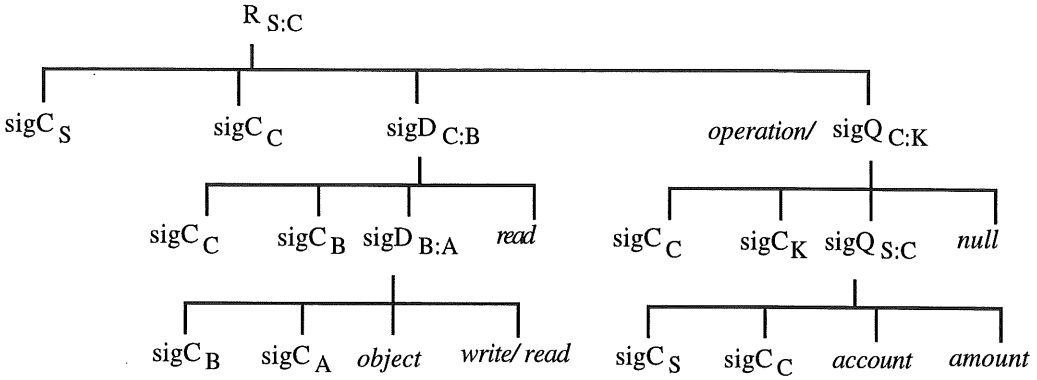
5. Using SAProxies for Non-repudiation

We now illustrate how SAProxies may be used to provide non-repudiation. The SAProxy mechanism itself does not enforce mandatory non-repudiation. Instead, SAProxies provide the means by which both POO and POR may be enforced at the discretion of the participants involved in a transaction. It allows principals to define their own working policies, and to set limits on the risks they are prepared to take. We consider the use of SAProxies to solve issues of non-repudiation where a resource or service is provided and must be paid for. This situation requires that the request for the service is genuine and can be proved to have occurred, that the service has indeed been provided and that where it is received it must be paid for and not denied.

As all SAProxy requests for access to resources or information are attributable, the provider of a service has no problem proving that his services have been demanded. He can then use the requests to generate invoices to demand payment. Payment may then be effected by sending an SAProxy cheque, as shown above. By the same argument, a service provider cannot claim that a client has used his services without producing a valid request SAProxy signed with the clients private key.

However, once a service has been provided, it is not possible to withdraw it, so first of all we consider that the service provider may want to ensure that the requester can pay for the service. This is of particular importance with remote clients that are unknown. The service provider may, if the sums of money involved are small, be prepared to just hope that a cheque will arrive (the client may not deny receiving the service) and that it will not bounce. SAProxy cheques can be guaranteed as shown above. The service provider may wish to have a cheque up front, when the request is made. The client may also want to ensure that the service provider cannot deny that he has received the cheque for that particular transaction. These situations may be resolved by binding the cheque SAProxy with the request for service.

The following tree shows a request from C to the server S. C not only 'delegates his access right' to operate on the object to S, but also gives S access to the appropriate amount of money from his account.



The client must also receive the service and not be invoiced for a service he has not received. A possible solution to this may be found if the client agrees his own terms of operation with his bank. For example, C may arrange that his cheques to a particular provider are not cashed until his request, $R_{S:C}$, is returned to the bank having been endorsed, with another SAProxy, by himself.

Finally the client must not be able to deny receipt of the service. Some form of POR of the service is required. This is a different situation to that dealt with in [2]. Whereas a NRS can withhold a signed message, data, until POR has been negotiated, it would have no means of controlling the provision of a service or resource. This is the most difficult situation to deal with, as it appears to be dependent on the honesty of the client. We do not, as we have said previously, advocate that a fixed mandatory mechanism be followed, we feel that this would be impractical. However, SAProxies provide a wide scope of solutions each of which may be tailored to the different requirements of the parties involved. Where POR is an absolute must, and can not rely on the honesty of a single interested party, then as in [2] a third party, trusted by provider and client, must be involved.

A possible solution may be for this third party to 'witness' receipt of the service and then to endorse the request, $R_{S:C}$, when the service has been received in order to prove that the service was provided. The third party, T, would issue an SAProxy, $I_{K:T}$, to the bank (or to the service provider, depending on the policy) to indicate that it has ascertained that the service had been provided.

$I_{K:T}$: $sigC_K$ $sigC_T$ $R_{S:C}$ *null* $sigI_{K:T}$

The bank may then cash the cheque or an arbitrator may then enforce payment of the cheque. The advantage of this third party over an NRS is that this need not be a global authority. Instead it may be a party, without a specific function, which all principals involved in the transaction are able to accept in that role.

Most business transactions, however, require simplicity of operation for the sake of efficiency and, in general, principals, are prepared to accept a certain amount of risk. The level of risk depends on the trust placed on the other party and on the amount of loss that may be sustained. Typically, these third parties are not necessary, instead business transactions have some dependence on the honesty of the participants. Where a participant fails to comply with the agreed policy, then the general solution is to cut one's losses and have no further dealings with them. Thus a variety of solutions may be implemented using SAProxies depending on the needs of those involved and who is to be trusted to what extent.

6. Summary and Conclusion

In this paper we have shown that non-repudiation is an issue not just in an exchange of data but also in sharing resources and in business transactions. We have described the scheme proposed by Coffey and Saidha and discussed its limitations. We have then proposed an alternative scheme, SAProxies, and we have illustrated how SAProxies can be used to provide non-repudiation in a particular situation. It is not hard to imagine how this flexible scheme may be

adapted to provide the same assurance in other situations.

We have not discussed the generation of PKCs, the role of certification authorities and the implications of invalid PKCs. We assume that each principal has a certification authority it can trust and that the PKCs of remote principals, including certification authorities, can be obtained in a manner similar to that used by PGP [7]. Although there is room for much discussion in this area we believe it is outside the scope of this paper.

We have shown that SAProxies can easily be adapted to suit the needs of the business world, so that the risks taken by both the provider and receiver can be reduced to acceptable levels in order to work together in much the same way that we do in ordinary every day life. Each participant can decide what risks to take, and set up own methods of operation to achieve whatever level of non-repudiation is necessary for their particular operation.

We conclude that the SAProxy scheme can be applied widely to provide non-repudiation in many situations. It allows participants in a transaction to enforce at their discretion either proof of origin, or proof of receipt or both. As SAProxy tokens are generated and verified locally, the scheme is easy to use in a global, distributed environment. There is no need for a globally trusted authority or hierarchy of authorities and therefore there is no need to accommodate the ramifications of using such an entity.

The purpose of a non-repudiation scheme should be to provide attribution of actions and not to control the behaviour of parties involved in an exchange. The mandatory non-repudiation scheme defined by Coffey and Saidha prevents misbehaviour of communicating principals as nothing takes place without the approval of the NRS. SAProxies do not necessarily inhibit misdeeds, but they may be employed so as not to allow any event to occur without leaving proof of who did it.

References

[1] Christianson B., Low M.R. Key-spoofing attacks on nested signature blocks. *IEE Electronics Letters*, 31(13):1043-1044, June 1995.

[2] Coffey T., Saidha P. Non-Repudiation with Mandatory Proof of Receipt *ACM Computer Communication Review*, 26(1):6-17, January 1996

[3] Diffie W., Hellman M.E. New Directions in Cryptography. *IEEE Transactions on Information Theory*, 22(6):644-654, November 1976.

[4] Low M.R., Christianson B. A Technique for authentication, access control and resource management in open distributed systems *IEE Electronics Letters* 30(2):124-125 January 1994

[5] Low M.R., Christianson B. Self Authenticating Proxies *Computer Journal* 37(5):422-428 October 1994

[6] Low M.R. *Self Defence in Open Systems: Protecting and Sharing Resources in a Distributed Open Environment* Hatfield: University of Hertfordshire, Computer Science Division. Thesis (PhD), September 1994.

[7] Stalling W. The PGP Web of Trust *BYTE* pp161-162, February 1995