Privacy and Trust in Distributed Networks

Thomas Rössler and Arno Hollosi

Institute for Applied Information Processing and Communications (IAIK),
Graz University of Technology, Inffeldgasse 16a, 8010 Graz, Austria
{thomas.roessler,arno.hollosi}@iaik.tugraz.at

Abstract. Today distributed service frameworks play an ever more important role. Transitive trust is of great importance in such frameworks and is well researched. Although there are many solutions for building and transmitting trust in distributed networks, impacts on privacy are often neglected. Based on a trust metric it will be shown why insufficient trust is eventually inevitable if a request or message pass through a chain of services. Depending on the reaction of the service, privacy critical information may leak to other entities in the chain. It is shown that even simple error messages pose a privacy threat and that proper re-authentication methods should be used instead. Several methods of re-authentication and their impacts on privacy are discussed.

1 Introductional Example

The following example shall illustrate the problems concerning privacy and trust relationships in a distributed network of services. Here, in order to process a special task, the cooperation of several autonomous services is needed.

Assume the Following Situation. A client would like to buy something which is offered by an online shop where the client is registered in the customer data base. Therefore, the buyer signs a purchase order through the shop's online portal. The process involves several services of other instances such as a service for checking the inventory of the chosen product or a service for doing the payment. Figure 1 depicts the constellation of services used in this example.

After the user has been authenticated to the portal of the online shop (*service A*), he has to fill in some forms and enter some personal data. When this first step is completed, service A contacts the service which processes a new order (*service B*). This service validates and verifies the content of the order. First, it checks if the desired product is still available for sale. This will be done by sending a request to the inventory service (*service F*). Next, service B initiates the payment service (*service C*) to process the payment of the product. The payment is preferably done by the client's customer account which allows the client to buy products within a certain credit line. Thus, service C contacts a service of the internal account system (*service D*). Assuming, that the user has not enough money on his account, the payment service tries to make the payment by a direct debit to the client's credit card institution by requesting

A. Lioy and D. Mazzocchi (Eds.): CMS 2003, LNCS 2828, pp. 117–131, 2003.

the corresponding *service E*. At this point, some problems concerning privacy and transitive trust will arise:

- Because of too low security restrictions in the chain of services, it might be possible that service E does not accept the request.
- Depending on how service E reacts to this situation, private data about the client may be disclosed to other services and the client's privacy may be harmed.

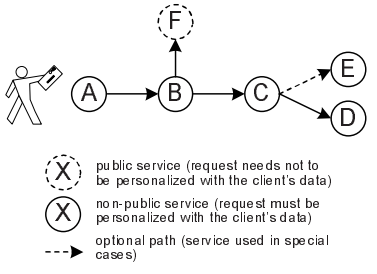

Fig. 1. Example of distributed services

If service E does not trust the request from service C, it is likely that an error message is sent back to the client. Passing this message back to the user through the service chain harms the user's privacy because each service learns about the error. In the worst case, the error message contains the information of its origin, service E. Thus, every service in the chain gains knowledge that the client has not enough money and needs to use his credit card. Otherwise it would not have been necessary to involve service E into the process. Moreover, by recording a client's habit it would be possible for instance to recognize that his customer account is overdrawn every end of month. Instead of sending an error message in response to insufficient trust into the authentication, service E can request a re-authentication of the client. Again, similar problems and privacy threats arise in this case. In the next section a trust model and applicable trust algebra is described, which is based on the work of Audun Jøsang ([7],[8],[9],[10]). Based on this a metric for determining the trustworthiness of a request inside a chain of services is discussed. By introducing and adapting trust values ([1],[2]) based on established criteria ([6],[4]) it is possible to decide either to trust or distrust an incoming request. This will lead to the necessity of re-authenticating requests if a service does not trust the incoming request. In the last section, privacy and threats on privacy are described. Error messages and re-authentication requests are considered critical to privacy and so they are discussed in detail at the end of this paper.

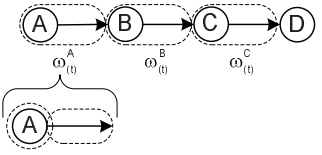

Fig. 2. An example of chained trust

2 The Opinion Triangle

2.1 Definitions

Initially, the trustworthiness of each service has to be determined. Therefore, every entity can be divided in two parts. On the one hand, the connection between two services has to be evaluated under the aspect of security. For example, a normal TCP/IP connection will result in a lower level of security than an SSL-connection with client certificates. On the other hand, the service itself has to be evaluated. For this purpose, some established criteria already exist, e.g. the *Common Criteria* [6]. Such criteria not only consider the technical infrastructure and the system itself. They also take the technical and nontechnical environment into account. After evaluating each service, the level of trust must be expressed in an applicable metric. Therefore, in [10] and [9] the term *opinion* (ω) was defined as:

$$t + d + u = 1, \{t, d, u\} \in [0, 1]^3 \tag{1}$$

Definition Opinion: *Let $\omega = \{t, d, u\}$ be a triplet satisfying (1) where the first, second and third component correspond to trust, distrust and uncertainty respectively. Then ω is called an opinion.*

Corresponding to this definition, several levels of trustworthiness are mapped accordingly to different opinions. Equation 1 defines a triangle which is depicted in figure 3. Every opinion ω can be described as a point $\{t, d, u\}$ in the triangle. For example, there are five trust levels to distinguish (according to table 1) and each trust level can be found in the opinion triangle (fig. 3).

Table 1. Example of mapping between trust levels and opinions $\omega = \{t, d, u\}$

t	d	u	trust level	
0.00	0.95	0.05	distrust	(-1)
0.10	0.10	0.80	ignorance	(0)
0.40	0.10	0.50	minimum trust	(1)
0.70	0.15	0.15	medium trust	(2)
0.95	0.00	0.05	maximum trust	(3)

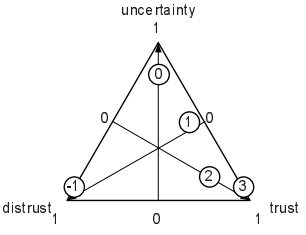

Fig. 3. Trust levels inserted into the opinion triangle

The advantage of using the opinion based trust model instead of a simple trust level based model is that there are three parameters expressing trust instead of only one value. As it will turn out in the next section, these three separate values are not treated equally when different opinions have to be combined. This will result in a real world adequate model for distributed trust relationships.

3 Subjective Logic

The algebra for determining trust will be based on a framework for artificial reasoning called *Subjective Logic*, which has been described in Audun Jøsang's papers [8] and [9]. It defines various logical operators for combining opinions. In this section, only the most important definitions will be quoted, e.g. the *Recommendation* and *Consensus* operator. Finally, the subjective logic allows to examine joined entities under the aspect of trust.

3.1 Definition: Conjunction

If some entity has two different opinions about another entity, then the conjunction ($\wedge$) of these opinions may be useful.

Let $\omega_p^A = \{t_p^A, d_p^A, u_p^A\}$ and $\omega_q^A = \{t_q^A, d_q^A, u_q^A\}$ be entity A's opinions about two distinct binary statements p and q. Then the conjunction of ω_p^A and ω_q^A, representing A's opinion about both p and q being true is defined by [9]:

$$\begin{aligned} \omega_{p\wedge q}^A &= \omega_p^A \wedge \omega_q^A \\ &= \{t_{p\wedge q}^A, d_{p\wedge q}^A, u_{p\wedge q}^A\} \end{aligned} \tag{2}$$

where

$$\begin{aligned} t_{p\wedge q}^A &= t_p^A t_q^A, \\ d_{p\wedge q}^A &= d_p^A + d_q^A - d_p^A d_q^A, \\ u_{p\wedge q}^A &= t_p^A u_q^A + u_p^A t_q^A + u_p^A u_q^A. \end{aligned} \tag{3}$$

3.2 Definition: Recommendation

Recommendation ($\otimes$) is needed if an entity A decides about the trustworthiness of something (p) based on trust-recommendations given by a third party B. More formally:

Let A and B two entities where $\omega_B^A = \{t_B^A, d_B^A, u_B^A\}$ is A's opinion about B's recommendation, and let p be a binary statement where $\omega_p^B = \{t_p^B, d_p^B, u_p^B\}$ is B's opinion about p expressed in a recommendation to A. Then A's opinion about p as a result of the recommendation from B is defined by [9]:

$$\begin{aligned} \omega_p^{AB} &= \omega_B^A \otimes \omega_p^B \\ &= \{t_p^{AB}, d_p^{AB}, u_p^{AB}\} \end{aligned} \tag{4}$$

where

$$\begin{aligned} t_p^{AB} &= t_B^A t_p^B, \\ d_p^{AB} &= t_B^A d_p^B, \\ u_p^{AB} &= d_B^A + u_B^A + t_B^A u_p^B. \end{aligned} \tag{5}$$

It must be mentioned that ω_p^B is actually only the opinion that B recommends to A and it is not necessarily B's real opinion. The opinion about an entity's recommendation, e.g. ω_B^A, results of the conjunction of two separate opinions. On the one hand, there is entity A's opinion about the trustworthiness of entity B by itself, called $\omega_{KA(B)}^A$. On the other hand, entity A's opinion about the trustworthiness of the recommendations (recommendation trust) made by entity B has to be considered, given as $\omega_{RT(B)}^A$. Applying the conjunction operator as defined above results in:

$$\omega_B^A = (\omega_{KA(B)}^A \wedge \omega_{RT(B)}^A) \tag{6}$$

This term is also known as the *conjunctive recommendation term* [9].

3.3 Definition: Consensus

The consensus ($\oplus$) operator is used to combine several independent opinions about the same statement. As a result the certainty should increase.

Let $\omega_p^A = \{t_p^A, d_p^A, u_p^A\}$ and $\omega_p^B = \{t_p^B, d_p^B, u_p^B\}$ be opinions respectively held by entities A and B about the same statement p. Then the consensus opinion held by an imaginary entity [A,B] representing both A and B is defined by [9]:

$$\begin{aligned} \omega_p^{A,B} &= \omega_p^A \oplus \omega_p^B \\ &= \{t_p^{A,B}, d_p^{A,B}, u_p^{A,B}\} \end{aligned} \tag{7}$$

where

$$\begin{aligned} t_p^{A,B} &= (t_p^A u_p^B + t_p^B u_p^A)/(u_p^A + u_p^B - u_p^A u_p^B), \\ d_p^{A,B} &= (d_p^A u_p^B + d_p^B u_p^A)/(u_p^A + u_p^B - u_p^A u_p^B), \\ u_p^{A,B} &= (u_p^A u_p^B)/(u_p^A + u_p^B - u_p^A u_p^B). \end{aligned} \tag{8}$$

The effect of the consensus operator is to reduce the uncertainty. Opinions containing zero uncertainty can not be combined.

Equipped with these three basic operation, it is possible to form a model for determining distributed trust in the web service scenario.

4 Chained Trust

As a basis for any calculation, each service must already have assigned an opinion ω about its security level—preferable by an independent authority. This opinion will be determined initially, during setting up the service, and has to be kept up-to-date. With some precautions, for example wrapping the opinion value into a signed certificate, the trust value could be sent within the requests. Anyway, trust values, opinions about the trustworthiness of an entity, respectively, have to be propagated in the network.

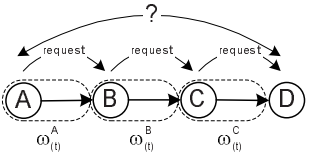

Fig. 4. The problem of chained trust

Figure 4 depicts the stated problem: a request originating from service A will be propagated through service B, C to D. Because of the security requirements of service D, there must be a mechanism to decide whether the request is trustworthy or not. This question is similar to determining service D's opinion about the trustworthiness of service A. Because of the indirect relationship between service A and D, the principle of recommendation is used. Let us consider the chained situation step by step. At first, service B has to decide about the trustworthiness of service A. This can easily be done by evaluating the opinion of service A's trustworthiness ω_t^A, which preferable was attached to the request. Because of the direct trust relationship between A and B, ω_t^A is the value which enables

a decision. In the next step, assuming that service B considers A's request as trustworthy, service B sends a request to service C. At this point, service C has to decide whether to trust or distrust the whole chain. Therefore, the subjective logic is needed. A direct trust relationship exists between B and C and the opinion ω_t^B is received by service C through the request. However, between service A and C there is no such direct relationship. In order to decide about the trustworthiness of the chain, respectively about the trustability of service A, the recommendation operator is applied. In this case, the opinion ω_t^A about service A's trustworthiness is recommended to service C by the preceding service B.

In the definitions stated by Audon Jøsang [9] there is a difference between the opinion ω_t^A about the trustability of an entity A and the opinion about recommendations of an entity. Therefore, the recommendation operator as introduced in section 3.2 requires the so called conjunctive recommendation term (equation 6), e.g. ω_B^A, which combines the opinion of the trustworthiness about a service itself and the opinion about its recommendation by applying the conjunction operator (as stated in 3.2). In this work, these two opinions are considered as equal. This assumption is legitimate in this context because in this scenario, if a service is not trustworthy and its trust value is respectively low, then the recommendations of this service should also be considered as not trustworthy and vice versa. Thus, the conjunctive recommendation term is built by the use of only one opinion and the term can be reduced to (equ. 9):

$$\omega_B^A = (\omega_{KA(B)}^A \wedge \omega_{KA(B)}^A) = (\omega_t^B \wedge \omega_t^B) = \omega_{t\wedge t}^B \tag{9}$$

Therefore, it is not necessary to define a separate opinion for recommendations which simplifies the application of the recommendation operator. With this assumption, the trust relationship at service C can be calculated as follows:

$$\begin{aligned} \omega_{t(A)}^{CB} &= \omega_B^C \otimes \omega_t^A \\ &= (\omega_t^B \wedge \omega_t^B) \otimes \omega_t^A \\ &= \omega_{t\wedge t}^B \otimes \omega_t^A \end{aligned} \tag{10}$$

In (10), C's opinion about the trustworthiness of service A consists of:

- the conjunction ($\wedge$) of C's opinion about B's recommendations and B's authenticity. In this matter, they are the same, namely ω_t^B.
- B's opinion about the trustworthiness of service A (ω_t^A)

With this result, service C is able to decide about the trustability of the chain. The same problem arises in the next step. Then, service D receives the request from the preceding service and it has to decide whether to trust or to distrust the chain of services. Based on recommendations as mentioned above, service D will calculate the opinion $\omega_{t(A)}^{DCB}$ in order to determine the trustability of service A through the chain of recommendations.

$$\begin{aligned} \omega_{t(A)}^{DCB} &= \omega_C^D \otimes \omega_B^C \otimes \omega_t^A \\ &= (\omega_t^C \wedge \omega_t^C) \otimes (\omega_t^B \wedge \omega_t^B) \otimes \omega_t^A \\ &= (\omega_t^C \wedge \omega_t^C) \otimes \omega_{t(A)}^{CB} \\ &= \omega_{t\wedge t}^C \otimes \omega_{t(A)}^{CB} \end{aligned} \tag{11}$$

The pattern of calculation is always the same. Moreover, it can be shown that this determination is recursive. With the opinion about the trustability of the chain so far, which was determined at the preceding service, and with the opinion about the trust relationship between the actual and the previous service, the chain can be evaluated. We will summarize this recursive approach to the calculation in the following lemma (12):

Lemma: Recursive Trust *Let* $A_1 \ldots A_n$ *be a chain of services, where service* A_{n-1} *makes some request to service* A_n. A_{n-1}*'s opinion about the trustworthiness of the chain so far is given by* $\omega_{t(A_1)}^{A_{n-1}\ldots A_2}$ *and it is attached to the request.* A_n*'s opinion about the trustworthiness of the whole chain is:*

$$\begin{aligned} \omega_{t(A_1)}^{A_n\ldots A_2} &= \omega_{A_{n-1}}^{A_n} \otimes \omega_{t(A_1)}^{A_{n-1}\ldots A_2} \\ &= \omega_{t\wedge t}^{A_{n-1}} \otimes \omega_{t(A_1)}^{A_{n-1}\ldots A_2} \end{aligned} \tag{12}$$

In consequence of this recursive calculation it is possible to evaluate the trustworthiness of the whole chain of services without having a particular list of all involved services. Such a list would be a threat for privacy as well. Tracing the components of the opinion about the trustworthiness of the whole chain during its propagation (based on recommendation) leads to the conclusion that the trust-component will never increase and the uncertainty generally becomes higher. Furthermore, at the end of the chain, depending on the particular opinions during the propagation, the uncertainty about a request may be too high for a service with sophisticated security restrictions. This is the reason why services either decline to act on the request and return an error message, or need the possibility to re-authenticate the client. In the next section we are going to look at the privacy threats that arise from this situation.

5 Privacy and Re-authentication

Insufficient trust leads to either declining a request or forcing a re-authentication. Webster's dictionary defines privacy as "freedom from unauthorized intrusion" which is also an adequate definition for this situation. Here, privacy means that the involved services should not gain more knowledge than it is necessary.

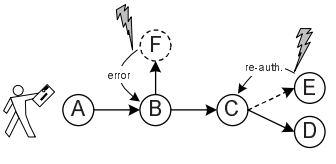

Fig. 5. Errors in distributed services

In the example given in the introduction, the error message disclosed enough information to conclude about the customer's financial situation. Therefore, er-

ror messages can act as side-channels. By analyzing similar processes initiated by different people it is possible to establish the standard workflow. But some client's request causes an error message or a re-authentication request due to a too low security level in the chain of services (service E in our example). With this information and with enough knowledge about the process it is possible to conclude about the involved services. In addition it is possible to gain information about user's request and about the user himself. This is why it is crucial to react carefully in such a situation. There are two possible reactions:

- replying with an error message
- starting a re-authentication procedure

Chain History. The question arises how and when information is propagated in order to reach previous services or the original client directly. One possibility is adding a chain history of preceding services to every request. The benefits of this approach are obvious: the original client is known to every service and each service can decide to trust the request based on the history of the request instead of calculating a level of trust. But adding a history of all involved services to every message not only increases the header of such messages, it also harms the user's privacy. Every involved service learns about all preceding services. In our example, service E (credit card institute) would learn about the user's contact to the online shop portal represented by service A, and that the user is going to buy something but does not have enough money (service B and service C respectively.) It seems that harming privacy is a too high price for the benefits of a chain history. Thus, a request or message should contain information of the sending and the receiving service only. No service should get more information about the process automatically, or more generally: a service should get as much information as needed but not more than absolutely necessary.

5.1 Error Messages

The minimum reaction is to return an error message to the preceding service. In the situation depicted in fig. 5, service E rejects the request from service C due to security considerations. Therefore, service C will receive an according message. Depending on how detailed the error message is, the receiving service will react. In the worst case, the error will be reported backwards through the whole chain to the original client. On the one hand, in order to give the user as much help as possible, the error message should be very detailed. On the other hand, a detailed error message, which in the worst case passes through every entity in the chain, gives all desirable information about the whole process and the user himself. For this reason, such messages should be encrypted with the client's public key. This prevents disclosing detailed information to any third party. But already the occurrence of an error message brings enough information. Nevertheless, there must be at least a message about the unsuccessfully terminated process that has to be sent to every involved service in order to stop the process. It is preferable to use a solution where services try to fulfill the request without rejecting an

error message. The usage of re-authentication is an attempt to do so. In same cases it may be the only practical way in a distributed service framework.

5.2 Re-authentication Requests

A re-authentication request is sent to the user in order to authenticate the request for a dedicated service. It is also possible that instead of the user himself a trusted third party is allowed to sign the request on behalf of the user. A re-authentication request has to contain at least the following data:

- a pseudo-random stream or a digital finger-print of the sending service (hash-value)
- a time-stamp or nonce to prevent replay-attacks
- an explanation of the receiving service and the purpose of this service in plain text (readable for the user who has to sign it)
- a signature over the request with the private key of the sending service in order to prove the origin of this request

The time-stamp or nonce is needed to prevent manipulation of a service with a replayed re-authentication request in order to gain confidential information about a client. This component is essential for security and is common practice in security technologies. The additional text of the request has to contain detailed information about the service which wants the user to re-authenticate himself. The user must be able to recognize the circumstances of this request in order to decide correctly. Furthermore, the explanation in the request must point out the consequences and results of signing and executing the request. At last, the whole re-authentication request has to be signed by the sending service in order to prove the origin of the message. With such a detailed request, the client will be well informed and will be able to decide whether to grant the permission by re-authenticating or not to grant permission.

Re-authentication requests can be split up into synchronous and asynchronous requests. In a synchronous request, the re-authentication request has to be fulfilled in time. That means that the requesting service is waiting until the request is sent back. This is a viable option only if it can be assumed that this will happen within a certain time frame. Contrarily, the asynchronous request leads to a temporary interruption of the process, because it is not predictable when the re-authentication request will be sent back. If too much time elapsed between starting the process and answering to the re-authentication request, some problems may arise concerning time restrictions for some outstanding requests in the chain. Therefore, a re-authentication through a synchronous way should be the first choice. The re-authentication can be carried out by four means, as follows.

Out-of-Band Re-authentication. When using this method, every service contacts the user directly. It implies the requirement that every service has to get information about how to reach the user in an out-of-band way. Therefore,

it is necessary to add some information like the client's e-mail address to the request.

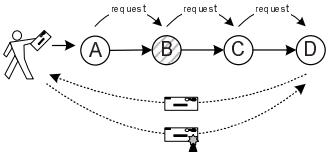

Fig. 6. An example of an out-of-band re-authentication

From the point of view of privacy this is the best solution, as services do not gain any more information by adding (possibly temporary) contact information to the requests. And in case of errors or re-authentication no information is disclosed to other services in the chain (see figure 6.) Assuming that the client signs the request immediately, no other service in the chain will recognize the re-authentication. The drawback is that a re-authentication in this way will most likely be asynchronous (for example using e-mail). It is not very likely for all users to have their own server running which services can contact for re-authentication.

Roll-Back Re-authentication. Using this method the re-authentication request is passed step by step back to the client. Beginning from the last service, e.g. service D, a re-authentication request will go through the whole chain backwards until a trustworthy service or the user itself is reached. The request is then signed and sent back. (fig.7). Each entity in the chain will learn that something is going on, and depending on the content of the request private information may be disclosed.

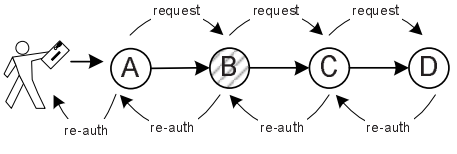

Fig. 7. Roll-back re-authentication

Using this method it is very important to encrypt the content of the re-authentication request. Otherwise, every involved service which transmits the request will gain additional information about the process. But even if the content of the request is encrypted and the identity of the requesting service is masked through some session-id privacy is threatened. In our introductory example, when service E issues service A an encrypted re-authentication request, service A can reason that the request originated from service E, as no other

service in the workflow would issue such a request. Thus, service A learns that the user has to have some problems in context with his financial situation. From the aspect of privacy roll-back re-authentication is not the best solution.

Contrarily to the out-of-band mechanism, this re-authentication is a synchronous possibility to get in contact with the user. This is because, the client is already logged in to service A and the re-authentication request is eventually presented to the user through service A.

Ticket-Server Solution. Similar to MS-Passport or the Kerberos authentication system ([12],[11]), this solution uses an additional ticket server (TS). As depicted in figure 8, in parallel to accessing the agent's portal (service A) the user signs in at the ticket server. Whenever an authentication is needed the service in question contacts the ticket server.

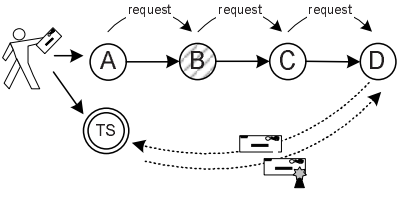

Fig. 8. Re-authentication by using a ticket server

After the user is successfully signed in at the ticket server, the ticket server is allowed to perform re-authentication requests by signing these requests on behalf of the user. Therefore, the server replies with tickets to the requesting services. The ticket itself is signed by the ticket server using its private key. In order to prevent unrelated services requesting an authentication ticket from the server, the server has to generate a session-id which the user will tie to his request to service A by using cryptographic techniques. Otherwise the ticket server would have to make plausibility checks on the re-authentication requests which is impractical in most but the trivial cases. Furthermore, the ticket server would need more information than a session-id. This in turn may create new privacy problems.

The main advantage of this solution is that in the case of a required re-authentication no other service will learn about it. Also, the ticket server itself has no idea about the other involved services which do not require re-authentication or the whole process as such. Moreover, this solution is very comfortable for the user because he is not burdened with the re-authentication. This is why there will be no additional time-delay caused by the client while answering the request. And so, this re-authentication can be made synchronous. The drawback is that the user has to have absolute trust in the ticket server itself. After all, the ticket server acts on his behalf. Therefore this server must be maintained by a trustworthy independent party. Of course, such a server will be a prime target for attacks.

Communication Server Solution. This solution is similar to the ticket server solution. However, here the so-called communication server (CS) does not act on behalf of the user but is only used as contact and communication point (fig.9). This scenario does not suffer the drawbacks of the previous solution.

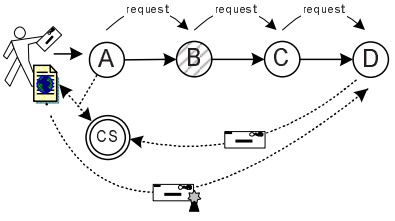

Fig. 9. Re-authentication through a web-server

The user contacts the CS and is reachable through the CS during the time of the process. For example, if the services have web-interfaces the communication server can be a special website. If a service wants to communicate with the client, e.g. because a re-authentication request is needed, it sends a request to the communication server. This implies that the services need to be aware of the IP-address of this web server. Thus, its address has to be propagated within the requests. Beside the IP-address, the requests may also contain some session-id in order to make it easier for the communication server to classify incoming requests. Both information do not provide additional private information about the user and are thus not privacy critical. The communication server passes on the request to the user who then signs the re-authentication request. The signed response is sent back directly to the requesting service.

This solution is similar to the out-of-band re-authentication. Here, the request is sent to a communication server instead of the client's mailbox. The request will be displayed directly through for example a web-page and the client can sign the request immediately. Therefore, the re-authentication is synchronous which is the difference to a common out-of-band solution. Apart from this difference, the content of the re-authentication request itself will be quite similar to the other solutions. The communication server could also be used to inform the client about the actual status of the process or to send him error messages. Generally, the communication server allows to communicate with the client without harming his privacy. The only precondition is that the communication server is trustworthy and is run by some reliable party.

5.3 Practical Aspects

The models and problems described in this section are crucial for applications in an e-governmental environment. Here too, distributed services are used to process transactions initiated by a client. Furthermore, because of the sensitive data involved with governmental transactions, protecting the client's privacy is of paramount importance.

From the discussed solutions in the previous section, out-of-band re-authentication cannot be used as some services require a synchronous re-authentication possibility. Roll-back re-authentication discloses too much information and should not be used in a privacy sensitive environment. That leaves the ticket server and communication server methods as options. However, such a server acts as central authentication authority for whichever governmental service the user accesses. Thus, such a server could be used to profile the user's actions. In addition, data protection laws may forbid running such a service in the context of governmental processes. How can this situation be resolved? Depending on the circumstances three solutions exist:

1. **using a private communication server**
2. **using different authentication authorities**
3. **minimize the need for re-authentication**

Ad. 1). If it can be assumed that the user has access to a private communication server (possibly run by a third party), this server can be used to solve the problem. Again, this is a central approach, but the point is that this server is unrelated to the accessed services and it is a conscious decision on the user's part to use and trust that server.

Ad. 2). The second solution is using many different authentication authorities, instead of only one. For example, the first service in a process could act as an authority. Furthermore, the user might choose different authorities for accessing different services as it is intended by the Liberty Alliance specifications [5] and the federated Single Sign-On approach [3]. This stands out against a central approach clearly, but has the drawback of possible privacy violations mentioned earlier.

Ad. 3). The third solution is to minimize the need for re-authentication. This can be achieved by digitally signing the request and binding it to the current session. The signature can be verified by every service in the chain and thus the user can reliably be authenticated at each service. To prevent sending the complete request and disclosing too much information to each service the request could be split into separate signed parts. Alternatively, the parts could be partially encrypted with the targeted service's public key. However, signing and splitting the request is not possible in all cases.

To sum up, every solution has its benefits and drawbacks and should be applied according to the situation at hand. Based on our experience, a sensible combination of the proposed solutions may solve most problems.

6 Conclusion

In this paper, we analyzed privacy threats in distributed service frameworks. First, we introduced a metric to calculate security in such frameworks. The term opinion was defined and an algebra which is based on recommendation

relationships was given. With this it is possible to determine the trustworthiness of a request without having information of all involved services. Determining trust stepwise through a chain of services led to the conclusion, that the trustworthiness of a request will decrease while the value of uncertainty will become higher. Thus, eventually a request may be considered as not trustworthy. In order to complete the request successfully, the necessity for some re-authentication mechanism arises. Furthermore, we have shown how the client's privacy can be harmed by simple error messages. The mere fact of the existence of error messages combined with a knowledge of the workflow may disclose private information to others. Next, several methods of re-authentication and their impact on privacy have been discussed. A consequence of this analysis is that messages should contain only the absolute minimum information necessary for the services to function correctly. Any more data may harm the client's privacy. Furthermore, encryption should be used wherever sensible, so that information can be passed to services further down the chain without disclosing it to intermediate nodes. It has also been shown that introducing a trusted third party may have substantial benefits from the point of view of privacy. We hope, that other researchers follow suit and consider privacy when designing methods for distributed service networks.

References

1. A. Abdul-Rahman and S. Hailes: A distributed trust model. In Proceedings of the New Security Paradigms 97, 1997.
2. A. Abdul-Rahman and S. Hailes: Supporting trust in virtual communities. In Proceedings of the Hawaii Int. Conference on System Sciences 33 , Maui, Hawaii, 2000.
3. J.D. Beatty et al: Liberty Protocols and Schemas Specification 1.0. Liberty Alliance, 2002.
4. ECSC-EEC-EAEC: Information Technology Security Evaluation Criteria (ITSEC), 1991.
5. J. Hodges et al: Liberty Architecture Overview 1.0. Liberty Alliance, 2002.
6. International Standardization Organisation (ISO): Evaluation criteria for IT security (ISO/IEC 15408:1999), 1999.
7. A. Jøsang: The right type of trust for distributed systems. In C. Meadows, editor, Proceedings of the 1996 New Security Paradigms Workshop, 1996.
8. A. Jøsang: Artificial reasioning with subjective logic. In Abhaya Nayak, editor, Proceedings of the Second Australian Workshop on Commonsense Reasoning, 1997.
9. A. Jøsang: An algebra for assessing trust in certification chains. In J.Kochmar, editor, Proceedings of the Network and Distributed Systems Security (NDSS'99) Symposium, 1999.
10. A. Jøsang: Trust-based decision making for electronic transactions. In L.Yngstrm and T.Svensson, editors, Proceedings of the Fourth Nordic Workshop on Secure IT Systems (NORDSEC'99), Stockholm, Sweden, 1999.
11. J. Kohl and C. Neuman: The Kerberos Network Authentication Service (V5). RFC 1510, 1993.
12. Microsoft Corporation: Microsoft .NET Passport – Technical Overview, 2001.