DISALLOWING UNAUTHORIZED STATE CHANGES OF DISTRIBUTED SHARED OBJECTS

J. Leiwo, C. Hänle, P. Homburg and A.S. Tanenbaum
Vrije Universiteit, FEW, De Boelelaan 1081A, 1081 HV Amsterdam, The Netherlands
{leiwo,chris,philip,ast}@cs.vu.nl

Abstract

Attaching digital signatures to state update messages in global distributed shared object (DSO) systems is not trivial. If the DSO consists of a number of autonomous local representative that use open, public networks for maintaining the state consistency, allowing a local representative to sign state update messages is not appropriate. More sophisticated schemes are required to prevent unauthorized state updates by malicious local representative or external parties. This paper examines the problem in detail, compares a number of possible solutions, and identifies the most suitable one and demonstrates how the state update messages can be signed using the identified solution.

1. INTRODUCTION

Assume a distributed shared object (DSO) consisting of a number of local objects (representatives), i.e. components that reside in a single address space and communicate with other local objects in different address spaces.

To use the DSO for, say, delivering digital products (e.g. software packages), it is meaningful to structure the DSO so that the authority to update the state of the DSO is only granted to an administrator accessing a limited number of trusted core local objects. The updated state is then propagated, according to a particular replication policy, to a larger number of less trusted caching local objects to which clients can bind to and download the product of interest.

A number of schemes have been developed to provide authorization in distributed object systems. Access control, however, is not enough. Each caching local object must verify the authenticity and integrity of the state they receive as a result of a DSO state update.

Using traditional digital signatures to authenticate the state update is straightforward. An administrator of a software package authorizes an update. The local object of the core group attaches a digital signature to the state update message and the cache group member verifies the signature using the core group member's public key. Evidence is provided to the client of the authenticity and integrity of the software package when downloaded by the client.

A number of factors reduce the applicability of this scheme in DSO systems. More advanced signature schemes shall be surveyed and the most suitable scheme selected for implementation in the software distribution scenario.

2. PROBLEMS OF STATE UPDATE SIGNING

The number of cache group members can be very large. Communication lines may be untrusted, and there may not be a secure cache group member establishment procedure. Because of global range, security solutions must be based on public key cryptography. However, even in the presence of a global PKI, the trustworthiness of certificates remains questionable [3].

Some PKIs do not require a globally trusted root certificate server. SPKI [2] allows creation of a local name space within which the certificates are valid. Certificate does not necessarily bind a name to a public key but a public key to an authorization. Because object references are not necessarily unique or persistent, identifying a local object by a public key simplifies the key management in dynamic environments.

In DSO applications with restricted user space (e.g. medical database), and assumption of a central authority is possible, as are hardware tokens to store public keys. However, in the digital product distribution example, this is not possible.

Further, the authenticity of the state should be bind to the identity of the human user that has authorized the state update. Administrator must convince users downloading a digital product from a cache group member that he (or the organization [s]he represents) has indeed authorized the state update propagated by the core group member.

DSO state may not be maintained locally. Transporting the possible very large state to the administrator workstation for signing introduces a significant communication overhead. The core group member could only transport a hash value of the state to be signed. However, the administrator should verify the hash value before signing, hence be in the possession of the full state.

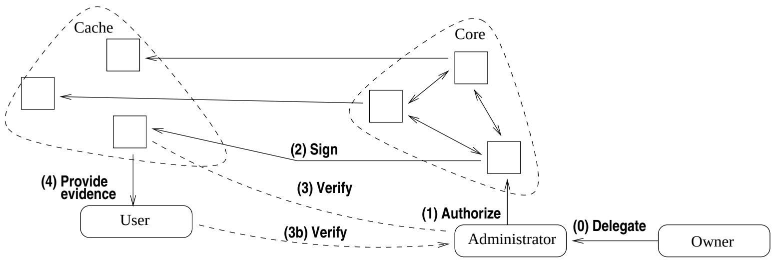

Figure 1 Distribution of free software using global distributed shared objects

Some DSO platforms, e.g. [9], allow per-object replication policies that may delay state updates. Authenticity and integrity of the state can therefore not be based on the assumption of the availability of the administrator to sign the state at the time of propagation.

To achieve an appropriate secure solution, security requirements and design choices must be carefully evaluated. In the following, the software distribution example shall be further elaborated to identify the exact security needs of state updates.

3. AN APPLICATION SCENARIO

Software distribution scenario is illustrated in Fig. 1. Administrators are connected to members of the core group through a special interface. Administrator can be an individual person or a group of persons identified by a static name and a shared or threshold signature key. Core group members share secure channels between each other but the channel between core and cache is not trusted. Users bind to cache group members.

If the cache representative is trusted, a secure channel can be established between a client's local representative and that representative. Alternatively, the local representative alone is trusted at the client side.

The administrator may only be representing an object owner, that has delegated the right to carry out certain DSO operations on his behalf (step 0). The administrator then authorizes a member of the core group to perform a state update and propagate it to cache group members (step 1). The core group member, on behalf of the administrator, attaches a digital signature to the state update message (step 2). The signing key must hold information of identities of both the core group member that carried out the signing and the administrator that authorized the signing.

If the cache is trusted, signature verification (step 3) consists of the cache group member verifying that the core group member providing a state update truly is operating on behalf of the administrator. This verification must be bind to the identity of the administrator (or the object owner) to prevent falsification of the verification process.

If the cache is untrusted, the signed state shall be delivered to the client that verifies the signature (step 3b) using the public key of the administrator (or the object owner) that authorized the state update.

Finally, the cache group member or the local representative, depending on the trust model, provides evidence to the user of the state authenticity (step 4).

A number of requirements for above a delegation scheme can be identified. First, the security provided by the delegation scheme should not be weaker than the security provided by a direct signing of the new state by an updating agent. Of course, this delegation should not significantly reduce the system performance.

No other party than the signing agent, i.e. the entity signing the state on behalf of another entity, must be able to generate a valid signature for any message.

The verifier must become assured of the agreement between the original signer and the signing agent, i.e. no signing agent must be able to successfully create a signature of an unauthorized state and claim it to he authorized by the original entity.

The signing agent must not be able to create a signature that can not be verified to have been created by that particular agent. Signatures generated by the signing agent must be distinguishable from those generated by the original signer.

There are a number of schemes for the original signer to delegate the signing right to a signing agent, as shall be surveyed in the following section.

4. DELEGATION SCHEMES

There are three basic alternatives for the delegation of the authority to sign messages from an administrator to the member of a core group: full delegation, delegation by warrant, and partial delegation by proxy signatures. Delegation schemes based on threshold cryptography can be imagined, but as they are likely to introduce a significant computational and communicative overhead, they shall not be further studied herein.

Full delegation requires the original signer to send her secret key to the signing agent that signs the new state using that secret key. Obviously, this gives away the control of the administrator's secret key and leads to the loss of most required properties of the delegation, non-repudiation in particular. Therefore, it shall not be further considered.

Delegation by warrant (called also delegation by a signed token) can be achieved either by a delegate proxy or a bearer proxy [10, 8].

In the delegate proxy scheme, the original signer creates a signed token indicating the designed proxy signer. The proxy signer then attaches this token to the signed message to indicate authority to act on behalf of the original signer.

In the bearer proxy scenario, a warrant is composed of a message and the original signer's signature for a new public key. The new public key, together with the corresponding secret key, is generated to the proxy signer who signs a message using the new key pair and attaches the certificate of the new public key to the message.

Proxy signature based delegation [6] allows core group members to sign messages on behalf of the administrator while maintaining a number of delegation security requirements.

The two delegation alternatives are proxy-unprotected proxy signature and proxy-protected proxy signature. In the first, both original signer and proxy signer can create valid proxy signatures whereas in the latter, only the proxy signer can create valid proxy signatures.

The basic security properties of digital signatures (namely, unforgeaebility and verifiability) are maintained in proxy signatures. Additional security requirements are also preserved in different proxy delegation alternatives.

Proxy signatures have attracted a considerable research interest. Kim *et al.* [5] have extended the proxy signature scheme to provide partial proxy delegation with warrant and threshold delegation. A different threshold proxy signature scheme has been established by Zhang [11]. Gamage *et al.* [4] have established a scheme to combine proxy signatures with proxy encryption.

Delegation by warrant is an efficient and straightforward solution for the delegation problem. It has the further advantage of simplifying dealing with delegation life-times and right revocation. Furthermore, there are no restrictions on which signature schemes can be implemented.

The major problem with delegation by warrant is the need for additional signature or key generation. In fact, in our examples, there are likely to be more than one delegation, so there are a number of extra signatures or key generations required. However, if the original delegatee is a human being (e.g. object owner), then the key certification problem can be solved.

Proxy signatures offer a somewhat more sophisticated way of solving the delegation problem. They preserve the desired characteristics of digital signatures, and can be used for per-message delegation of signing right. As the research community has also adopted proxy signatures, they are also the desirable solution for the DSO state authentication problem.

5. IMPLEMENTING DELEGATION

We shall first demonstrate theoretically how proxy signatures can be used in the software distribution example by core group members to sign the updated state on behalf of the administrator. Implementation aspects of the scheme shall then be discussed. This is followed by the evaluation of security and performance.

5.1. THE SCHEME

Let p be a large prime, q a prime factor of $p-1$, and g an element $g \in Z_p^*$ of order q. Let the administrator's private key be a random element $x \in Z_q$. The corresponding administrator's public key is $y = g^x \mod p$.

The proxy digital signature can be generated by the core group member on behalf of the administrator, and verified by a member of the cache group member in five steps. In **(1) Proxy key generation** the administrator selects $k \in Z_q$ in random and computes $r = g^k \mod p$. The administrator calculates the proxy signature key $s = x + kr \mod q$. In **(2) proxy key delivery** phase the administrator delivers the pair (s, r) to the member of the core group using a secure channel.

In **(3) proxy key verification**, the core group member verifies that $g^s = yr^r \mod p$. If this condition holds, the pair (s, r) is a valid proxy signature key. If the condition fails, an error recovery procedure is triggered. In the **(4) signing by core group member**, the core group member that signs the update state m computes a signature S_p using the available digital signature scheme using the key s as a secret key. The signature is (S_p, r).

Finally, in the **(5) verification** phase the member of the cache group, upon receipt of the update state, verifies the proxy signature as according to the original scheme. The only difference is that the public key y of the administrator is replaced with the key $y' = yr^r \mod p$.

5.2. IMPLEMENTATION ASPECTS

As the state update to cache group members does not necessarily happen immediately, (1) through (3) may happen immediately after the administrator has executed a state updating method, even though there may be a delay until the state update is propagated.

Mechanisms for selecting k and for calculating r and s can be integrated into the administrative client software. The only interaction required from the administrator is to unlock the secret key x stored on a disk or a smart card.

As the core is trusted, a secure channel can be established between the administrator's client and the core group member. This secure channel must provide at least protection of confidentiality, integrity and authenticity of data. Authorization measures must be on place to prevent access to state modifying methods from other clients than valid administrators. Therefore, the state of a DSO can only change according to an authorized request.

Assume a logical distinction between the replication and distribution of the updated state of the DSO, and the marshaling and signing the new state. In the example system [9], a dedicated replication subobject is implemented to handle the replication.

Using the dedicated communication subobject, the replication object distributes the state to other local objects according to the consistency policy. The state is the state of the semantics subobject that marshalls the state and passes it to the replication subobject through the control subobject.

When dictated by a consistency policy, the replication subobject requests for the semantics subobject's marshalled state. At this stage, the semantics subobject calculates the proxy signature on behalf of the administrator using the secret key s received and verified earlier.

The receiving cache group members have to obtain the (certified) public key y of the administrator prior to being able to verify the proxy signature. This is handled during the binding procedure. In fact, recent extensions to DNS [1] support delivery of public keys within the part of name service that constitutes a part of binding protocol.

Implementation repositories, from where clients download the code to construct a local representative, must be trusted. Therefore, the public keys of administrators can be securely be delivered during the construction of the local object. Hence, the need for additional certification and on-line verification of the certificates is reduced.

5.3. SECURITY AND PERFORMANCE

As far as authors are aware, no sever attacks are known against proxy signature schemes. Therefore, we can assume that *unforgeability*, *proxy signer's deviation*, *verifiability*, *distinguishability*, *identifiability*, and *undeniability* hold.

From the DSO application point of view, unforgeability and verifiability are the most important. They provide assurance that no unauthorized party can masquerade as a core group member and replicate an inconsistent state to cache group members. In the software distribution example, this would enable implementation of trojan horses or other malicious elements to the software packages. In more security sensitive scenarios, such as banking applications, this would enable incorrect account details being distributed to clients.

Identifiability and Undeniability are essential for accounting purposes. Even if acting on behalf of the administrator, each core group member can be held individually accountable for signatures generated. This also helps identification of sources of security violations in case a core group becomes substituted by a trojan horse.

Security of the proxy signature scheme is not significantly weaker than the security of the underlying signature scheme. Therefore, the proposed scheme holds the above security properties as well as the security level of the underlying signature scheme.

For performance considerations, the scheme must be compared to the performance to the schemes where the marshalled state is either signed by the core group member, or is transmitted to the administrator for signing. As the latter practically impossible, only first scheme is compared against the proxy scheme.

After each state modifying method invocation, the administrator client and the core group member must perform steps (1), (2) and (3) of the signing right delegation. Generation of the proxy signature key s requires a single modular exponentiation and a single modular multiplication by the client.

For the delivery of the key, availability of the secure channel can be assumed, there is no need for calculating the performance penalty of establishing the channel.

The verification of the proxy signature key requires two modular exponentiation and one modular exponentiation by the core group member. The relative performance increase by the proxy key verification depends on the underlying signature scheme used [7, pp.451-462]. Both DSA and ElGamal signature schemes, signature generation requires less computations than signature verification.

Generation of the DSA signature requires one modular exponentiation, one modular inverse with a 160-bit modulus, two 160-bit modular multiplications, and one addition. The exponentiation, that can be estimated to take approximately 240 modular multiplications is the major cost. In ElGamal signature, the signature generation takes one modular exponentiation (i.e. ca. 240 modular multiplications), the extended Euclidean algorithm, and two modular multiplications.

The two modular exponentiation and one modular multiplication are a significant performance overhead. However, the increase in the total delay of the receipt of an updated change is not significant. Assuming a global DSO, the total cost of the wide area communication is likely to be the major cause of the reduction in performance.

The significance of the computations relative to the total cost of replication can be reduced by applying the calculations over elliptic curves instead of multiplicative groups. Discrete logarithm problem based signature schemes, such as ElGamal, can be modified to operate on elliptic curves.

This does not improve the cost of proxy computations relative the cost of signing, but can reduce the total cryptographic overhead relative to the total cost of DSO state updates.

5.4. POSSIBLE EXTENSIONS

There are a number of possible extensions to the chosen approach in addition to the above measures to improve performance. Mostly these extensions can be used for applications where higher security is required.

One possible extension is to use proxy threshold signatures [11]. This enables a consensus of a number of administrators to be enough to approve a DSO state alterations. Instead of a single administrator being allowed to alter the DSO state, using threshold cryptography a subset of all administrators is required for the state update.

Proxy signcryption [4] can be used for enhancing the proposed scheme with cost-efficient confidentiality. Instead of separately encrypting the state update prior to signing it, the computations can be integrated by the proxy extension to the digital signcryption to improve the efficiency of computations.

6. CONCLUSIONS

We have demonstrated and evaluated a means to authenticate the state alterations in the distributed shared objects. The significant improvement over simply digitally signing the state prior to transmitting it to the components of the DSO is that the signature verification can be bind to the public key of the DSO administrator that authorized and executed the state alteration. However, to add flexibility into the system operations, the signing capacity is securely delegated to the DSO component using proxy signatures.

The proposed scheme is equally secure as the scheme where the signing right is not delegated. The performance relative to the cryptographic processing is considerably but relative to the total cost of the DSO state update, less significant. Further optimizations, such as selection of a suitable signature scheme and use of elliptic curve cryptography, can further reduce the added overhead related to the total cost.

The scheme is flexible to enable future adaptations into the different secure needs of different application scenarios.

References

[1] D. Eastlake. Domain name system security extensions. IETF RFC2535, March 1999.

[2] C. Ellison, B. Frantz, B. Lampson, R. Rivest, B. Thomas, and T. Ylönen. SPKI certifiacte theory. IETF RFC 2693, September 1999.

[3] C. Ellison and B. Schneier. Ten risks of PKI: What you are not being told about public key infrastructure. *Computer Security Journal*, 16(1):1–7, 2000.

[4] C. Gamage, J. Leiwo, and Y. Zheng. An efficient scheme for secure message transmission using proxy-signcryption. In *Proc. 22nd Australasian Computer Science Conference, ACSC'99*, Australian Computer Science Communications Vol.21, Nr.1, Springer–Verlag, 1999.

[5] S. Kim, S. Park, and D. Won. Proxy signatures revisited. In *Proc. First International Conference on Information and Communications Security, ICICS'97*, LNCS 1334, Springer–Verlag, 1997.

[6] M. Mambo, K. Usuda, and E. Okamoto. Proxy signatures for delegating signing operation. In *Proc. 3rd ACM Conference on Computer and Communications Security*, 1996.

[7] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone. *Handbook of Applied Cryptography.* CRC Press, NY, USA, 1997.

[8] B. Neuman. Proxy-based authorization and accounting in distributed systems. In *Proc. 13th International Conference on Distributed Computing Systems*, 1993.

[9] M. Van Steen, P. Homburg, and A. S. . Tanenbaum. Globe: A wide-area distributed system. *IEEE Concurrency*, pages 70–78, January–March 1999.

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

[11] K. Zhang. Threshold proxy signature schemes. In *Proc. 1st International Information Security Workshop, ISW'97*, LNCS 1396, Springer–Verlag, 1997.