# Ownership in a SOA Ecosystem View

## Security Model

Security is one aspect of confidence – the confidence in the integrity, reliability, and confidentiality of the system. In particular, security in a SOA ecosystem focuses on those aspects of assurance that involve the accidental or malicious intent of other people to damage, compromise trust, or hinder the availability of SOA-based systems to perform desired capability.

Security

The set of mechanisms for ensuring and enhancing **trust** and confidence in the **SOA ecosystem**.

Although many of the same principles apply equally to SOA as they do to other systems, implementing security for a SOA ecosystem is somewhat different than for other contexts. The distributed nature of SOA brings challenges related to the protection of resources against inappropriate access, and because SOA embraces the crossing of ownership boundaries, the security issues associated with the movement of data and access to functionality become more apparent in a SOA ecosystem.

From a people perspective, any comprehensive security solution for a SOA-based system must take into account that people are effectively managing, maintaining, and utilizing the system appropriately. The roles and responsibilities of the users, and the relationships between them must also be explicitly understood and incorporated into a solution: any security assertions that may be associated with particular interactions originate in the people that are behind the interaction.

We analyze security in terms of the [social structure](#SocialStructure)s that define the legitimate [permissions](#Permission), [obligation](#Obligation)s and [roles](#Role) of people in relation to the system, and mechanisms that must be put into place to realize a secure system. The former are typically captured in a series of security policy statements; the latter in terms of security guards that ensure that policies are enforced.

How and when to apply these derived security policy mechanisms is directly associated with the assessment of the *threat model* and a *security response model*. The threat model identifies the kinds of threats that directly impact the messages, services, and/or the application of constraints. The response model is the proposed mitigation to those threats. Properly implemented, the result can be an acceptable level of risk to the safety and integrity within the SOA ecosystem.

### Secure Interaction Concepts

We can characterize secure interactions in terms of key security concepts [ISO/IEC 27002]: confidentiality, integrity, authentication, authorization, non-repudiation, and availability. The concepts for secure interactions are well-defined in several other standards and publications. The security concepts are therefore not explicitly defined here, but are discussed related to the SOA ecosystem perspective of the SOA-RAF.

Related to the security goals in this section, there may be significant security policy differences between participants in different ownership domains. It is therefore important that these security policies and security parameters are negotiated at the start of the relationship between systems of differing ownership domains, and also when policies change between these domains. As with other policy conflicts, this is not to say that every policy negotiation is a custom, point-to-point interaction. Rather, common mechanisms and policies should be well known and appropriately accessible so the negotiation can be efficient and lead to predictable conclusions. Unnecessary complexity does not lead to effective security.

#### Confidentiality

Confidentiality is concerned with the protection of privacy of [participants](#Participant) in their interactions. Confidentiality refers to the assurance that unauthorized entities are not able to read messages or parts of messages that are transmitted, and is typically implemented by using encryption. Confidentiality has degrees: in a completely confidential exchange, third parties would not even be aware that a confidential exchange has occurred. In some cases, the identities of the [participants](#Participant) may be known but the content of the exchange obscured. In other cases, only portions of sensitive data in the exchange are encrypted.

Different ownership domains may have policies related to encryption mechanisms and/or cryptographic protocols between consumers and providers, and such policies need to be negotiated and understood prior to any interaction.

#### Integrity

Integrity refers to the assurance that information has not been altered in transit, and is concerned with the protection of information that is exchanged – either from inadvertent or intentional corruption. Section 5.2.4 describes common computing techniques for providing both confidentiality and integrity during message exchanges.

#### Authentication

Authentication is concerned with adequately identifying actors in a potential interaction or joint action. Various mechanisms and protocols can be used to achieve this goal. A combination of **identifiers** (as discussed in section 3.2.4.1) and other attributes of an actor is typically used to achieve this. The set of attribute values that claim to identify a specific actor are matched against the set of reference values expected for that actor and that are maintained by some trusted authority. If the comparison results in a sufficient match, authentication has been achieved. Which specific set of attributes is considered an adequate basis for comparison will be context-dependent and specifying such sets is not within the scope of the SOA-RAF.

In addition to the concern of adequately identifying each actor involved in the interaction, there may also be a need to provide authentication information related to the subject that initiated an interaction involving the combination of intermediary actors in a service orchestration scenario. In such a case, consumers and services work *on behalf of* the initiator of the interaction, and there may need to be mechanisms in place to identify the interaction initiator. This concern is covered later in section 5.2.5.

Authentication merely provides an assertion that an actor is the person or agent that it claims to be. Of itself, it does not provide a ‘green light’ to proceed with the interaction – this is rather the concern of **authorization**, covered below.

#### Authorization

Authorization concerns the legitimacy of the interaction, providing assurance that the actors have permission to participate in the interaction. Authorization refers to the means by which a [stakeholder](#Stakeholder) may be assured that the information and actions that are exchanged are either explicitly or implicitly approved.

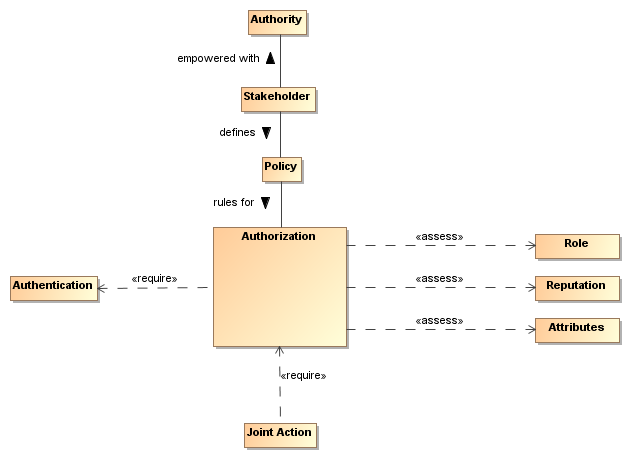


Figure - Authorization

The role of access control policy for security is to permit [stakeholder](#Stakeholder)s to express their choices. In Figure 41, such a policy is a written constraint and the role, reputation, and attribute assertions of actors are evaluated according to the constraints in the authorization policy. A combination of security mechanisms and their control via explicit policies can form the basis of an authorization solution.

The [roles](#Role) and attributes which provide a [participant](#Participant)’s credentials are expanded to include reputation. Reputation often helps determine willingness to interact; for example, reviews of a service provider will influence the decision to interact with the service provider. The [roles](#Role), reputation, and attributes are represented as assertions measured by authorization decision points.

#### Non-repudiation

Non-repudiation concerns the accountability of [participants](#Participant). To foster trust in the performance of a system used to conduct shared activities, it is important that the [participants](#Participant) are not able to later deny their actions: to repudiate them. Non-repudiation refers to the means by which a [participant](#Participant) may not, at a later time, successfully deny having participated in the interaction or having performed the actions as reported by other [participants](#Participant).

#### Availability

Availability concerns the ability of systems to use and offer the services for which they were designed. An example of threats against availability is a Denial Of Service (DoS) attack in which attackers attempt to prevent legitimate access to service or set of services by flooding them with bogus requests. As functionality is distributed into services in a SOA ecosystem, availability protection is paramount.

### Where SOA Security is Different

The distributed nature of the SOA ecosystem brings challenges related to the protection of resources against inappropriate access, and because the SOA paradigm embraces the crossing of ownership boundaries, providing security in such an environment provides unique challenges. The evolution of sharing information within a SOA ecosystem requires the flexibility to dynamically secure computing interactions where the owning social groups, [roles](#Role), and [authority](#Authority) are constantly changing as described in section 5.1.3.1.

Standards for security, as is the case with all aspects of SOA implementation and use, play a large role in flexible security on a global scale. SOA security may also involve greater auditing and reporting to adhere to regulatory compliance established by governance structures.

### Security Threats

There are a number of ways in which an attacker may attempt to compromise the security within a SOA ecosystem, primarily as attacks on the security concerns listed in section 5.2.1. The two primary sources of attack are (1) third parties attempting to subvert interactions between legitimate [participants](#Participant) and (2) entities that are participating but attempting to subvert other participants.

In a SOA ecosystem where there may be multiple [ownership boundaries](#OwnershipBoundary) and trust boundaries, it is important to understand these threats and protections that must be effective. Each technology choice in the realization of a SOA-based system can potentially have many threats to consider. Although these threats are not unique to SOA and can be mitigated by applying cryptographic techniques (digital signatures, encryption, and various cryptographic protocols) and security technologies, it is important that such threats are understood in order to provide solutions for thwarting such attacks and minimizing risk.

Message alteration

If an attacker is able to modify the content (or even the order) of messages that are exchanged without the legitimate [participants](#Participant) being aware of it then the attacker has successfully compromised the security of the system. In effect, the [participants](#Participant) may unwittingly serve the needs of the attacker rather than their own. Cryptographic mechanisms (hash codes, digital signatures, cryptographic protocols) can be used as a protection mechanism against alteration.

Message interception

If an attacker is able to intercept and understand messages exchanged between [participants](#Participant), then the attacker may be able to gain advantage. Cryptographic protocols can be used as a protection against interception.

Man in the middle

In a man-in-the-middle attack, the legitimate [participants](#Participant) believe that they are interacting with each other; but are in fact interacting with an attacker. The attacker attempts to convince each [participant](#Participant) that he is their correspondent; whereas in fact he is not. In a successful man-in-the-middle attack, legitimate [participants](#Participant) do not have an accurate understanding of the state of the other [participants](#Participant). The attacker can use this to subvert the intentions of the [participants](#Participant).

Spoofing

In a spoofing attack, the attacker convinces a [participant](#Participant) that he is another party.

Denial of service attack

A Denial of Service (DoS) attack is an attack on the availability and performance of a service or set of services. In a DoS attack, the attacker attempts to prevent legitimate users from making use of the service. A DoS attack is easy to mount and can cause considerable harm by preventing legitimate interactions in a SOA ecosystem, or by slowing them down enough, the attacker may be able to simultaneously prevent legitimate access to a service and to attack the service by another means. One of the features of a DoS attack is that it does not require valid interactions to be effective: responding to invalid messages also takes resources and that may be sufficient to cripple the target. A variation of the DoS attack is the Distributed Denial of Service (DDoS) attack, where an attacker uses multiple agents to the attack the target.

Replay attack

In a replay attack, the attacker captures the message traffic during a legitimate interaction and then replays part of it to the target. The target is persuaded that a similar interaction to the previous one is being repeated and it responds as though it were a legitimate interaction.

**False repudiation**

In false repudiation, a user completes a normal interaction and then later attempts to deny that the interaction occurred.

### Security Responses

Security goals are never absolute: it is not possible to guarantee 100% confidentiality, non-repudiation, etc. However, a well-designed and implemented security response model can reduce security risk to acceptable levels. For example, using a well-designed cipher to encrypt messages may make the cost of breaking communications so great and so lengthy that the information obtained is valueless.

Performing threat assessments, devising mitigation strategies, and determining acceptable levels of risk are the foundation for an effective process to mitigating threats in a cost-effective way.[[1]](#footnote-1) Architectural choices, as well as choices in hardware and software to realize a SOA implementation will be used as the basis for threat assessments and mitigation strategies.

#### Privacy Enforcement

The most efficient mechanism to assure confidentiality is the encryption of information. Encryption is particularly important when messages must cross trust boundaries; especially over the Internet. Note that encryption need not be limited to the content of messages: it is possible to obscure even the existence of messages themselves through encryption and ‘white noise’ generation in the communications channel.

The specifics of encryption are beyond the scope of this Reference Architecture. However, we are concerned about how the connection between privacy-related policies and their enforcement is made.

Service contracts may express confidentiality security policies and the cryptographic mechanisms required (e.g. ciphers, cryptographic protocols). Between ownership boundaries, there may also be similar security policies that define requirements for privacy between them. Between such boundaries, there may be a Policy Enforcement Point (PEP) for enforcing such requirements which may, for example, automatically encrypt messages as they leave a trust boundary; or perhaps simply ensuring that such messages are suitably encrypted in such a way as to comply with the policy.

#### Integrity Protection

To protect against message tampering or inadvertent message alteration, messages may be accompanied by the digital signature of the hash code of a message. Any alteration of the message or signature would result in a failed signature validation, indicating an integrity compromise. Digital signatures therefore provide a mechanism for integrity protection.

A digital signature also provides non-repudiation, which is an assurance of proof that a subject signed a message. Utilizing a digital signature algorithm based on public key cryptography, a digital signature cryptographically binds the signer of the message to its contents, ensuring that the signer cannot successfully deny sending the message.

The use of a Public Key Infrastructure (PKI) provides the support and infrastructure for digital signature capabilities, and there may also be security policies related to digital signatures between organizational boundaries, as well as trust relationships between multiple Certificate Authorities (CAs) across the boundaries.

#### Message Replay Protection

To protect against replay attacks, messages may also contain information that can be used to detect replayed messages. A common approach involves the use of a message ID, a timestamp, the message’s intended destination, signed along with the message itself. A message recipient may be able to thwart a message replay attack by

* checking to ensure that it has previously not processed the message ID
* validating that the timestamp is within a certain time threshold to ensure message freshness
* ensuring that the recipient is indeed the intended destination
* validating the digital signature, which provides non-repudiation of the message sender and checks the integrity of the message ID, timestamp, the destination, and the message itself, proving that none of the information was altered

Cryptographic protocols between participants can also be used to thwart replay attacks.

#### Auditing and Logging

False repudiation involves a [participant](#Participant) denying that it authorized a previous interaction. In addition to the use of digital signatures, an effective strategy for responding to such a denial involves logging of interactions and the ability to audit the resulting logs. The more detailed and comprehensive an audit trail is, the less likely it is that a false repudiation would be successful.

Given the distributed nature of the SOA ecosystem, one challenge revolves around the location of the audit logs of services. It would be very difficult, for example, to do cross-log analysis of services that write logs to their own file system. For this reason, a common approach revolves around the use of auditing services, where services may stream auditing information to a common auditing component which can then be used to provide interaction analysis and a common view.

#### Graduated engagement

Although many DoS attacks can typically be thwarted by intrusion detection systems, they are sometimes difficult to detect because requests to services seem to be legitimate. It is therefore prudent to be careful in the use of [resource](#Resource)s when responding to requests. If a known consumer tries to interact via a public interface that is not specified in the service contract, a service is not obliged to notice such an interaction request. In order to avoid vulnerability to DoS attacks, a service provider should be careful not to commit [resource](#Resource)s beyond those implied by the current state of interactions; this permits a graduation in commitment by the service provider that mirrors any [commitment](#Commitment) on the part of [service consumers](#ServiceConsumer) and attackers alike. A successful approach, however, cannot be implemented at the service-level alone – it involves a defense-in-depth strategy, coupling the use of intrusion detection systems, routers, firewalls, and providing the protections discussed in this section.

### Access Control

#### Conveying Authentication and Authorization Information

When an actor initiates an interaction with a service, that service may call other services or be part of a chain of service interactions as it carries out its functionality. Any service provider is aware of the immediate service consumer but, in some cases, for example, to provide proper access control to its data, a service provider may want information on who besides the immediate consumer is expected to see the data that is being requested. A significant question is whether trust of the immediate consumer should include trust that the immediate consumer will ensure proper data handling by its immediate consumer and back through any chain of service interactions. If this is not sufficient, conveying authentication and authorization information becomes a necessity, and the challenge becomes one of creating a conveyance process that gives more assurance than merely trust of the immediate consumer.. ~~As~~ ~~as well as the intermediaries~~ . This is a challenge both within and between ownership domains.

The security concerns related to conveying authentication and authorization information throughout intermediaries introduce significant complexity. Although an actor may directly authenticate to a service provider, that service provider may interact with other service providers in order to carry out its functionality, possibly without the knowledge of the initiator. There may therefore be privacy and confidentiality concerns related to conveying security information about the initiating actor. There may also be issues related to authorization, in that the initiating actor may need to explicitly delegate consent for intermediate services to act on the initiator’s behalf.

The following sections cover two approaches for conveying authentication and authorization information in a SOA ecosystem. These approaches involves conveying sufficient attributes, as discussed in section 5.2.1.3, which may be a single unique identifier or a set of identifiers that can be used in access control decisions.

In the first approach, the service consumer creates and passes an assertion about the initiating actor. In the second approach, a service is trusted to issue assertions about subjects. Each has specific implications for a SOA ecosystem.

##### Sender-Vouches Approaches

In a “sender vouches” approach, a service consumer creates an assertion, *vouching* for certain security information about the initiator of the interaction, and possible about other actors in a series (chain) of service interactions. This assertion contains sufficient attributes that can be used in access control decisions, and is sent, or propagated, to the service provider. Trust of such an assertion is therefore based on the provider’s trust of the consumer, and also there needs to be an understanding of such assertions between ownership boundaries. In a SOA ecosystem, such trust must be established at the beginning of each relationship.

When such assertions are reused in service orchestration scenarios beyond the initial consumer-provider interaction, there can be significant security risks[[2]](#footnote-2) .

* *Trust of Message Senders.* Because the trust of the assertion is based on the trust of the message senders, the more intermediaries there are, trust can degrade as the distance between the initiator and the service being called becomes greater. Trust may, therefore, be dependent on the trust of every sender in the chain to properly pass the claim.
* *Risk of Vulnerabilities in Intermediaries.* Because the trust of the assertion relies on the trust of each participant in the interaction, a risk is that intermediary services may become compromised and may inaccurately send false claims. Depending on the exact messaging syntax, an intermediary service could potentially manipulate the assertion or substitute another assertion. There could also be impersonation of the intermediary services, affecting the reliability of the interaction.

Approaches for mitigating risks in sender-vouches approaches involve a careful combination of SOA security governance, limiting the re-use of assertions beyond a certain number of points, establishing conditions of use for propagated assertions, keeping track of the history of the assertion in the interaction, and the use of digital signatures by an asserting party.

Between ownership domains, such an approach is even more challenging, as different ownership domains may recognize different authentication authorities and may not recognize identities from other organizations. Security policies that relate to the conveying of security information across boundaries must occur at the start of the relationship, with many solutions involving reciprocity of trust between authentication and authorization authorities from each domain.

##### Token Service-based Approaches

This approach revolves around use of a *token service* or a set of token services trusted to vouch for security information about authenticated actors in the interaction. In this approach, a token service issues a token which is an assertion that contains sufficient attributes that can be used in access control decisions. The service consumer passes this token, along with a request, to a service provider.

After the original consumer passes the issued token to the service, the recipient service later acting as a consumer may then choose to propagate the token to other service providers. Much like the risks associated with the reuse of assertions in sender-vouches approaches, there are risks associated with the reuse of tokens issued by the token service beyond the initial consumer-provider interaction. Most token service protocols and specifications, therefore, provide the capability for “refreshing” tokens for reuse in such situations. In this case, each actor retrieving a token may request that the token service issue a “refresh token” that can be propagated for a subsequent service interaction. Utilizing refresh tokens removes the risks associated with reuse.

This approach differs from the sender-vouches model in that trust of the token is not based on the message sender, but is based on the trust of the token service that issued it. In interactions between ownership domains, the establishment of the trust of the token services must be agreed to at the start of the relationship, and there must be an understanding of the policies associated with processing the tokens. To facilitate this, token services in one domain can often be used to “translate” tokens from other domains, issuing new tokens that are understood by services and consumers in its domain.

Unlike sender-vouches approaches, the token service approach revolves around a trusted token service or a set of trusted token services, and there may be architectural implications related to performance and availability. It is therefore advised that solutions that provide elastic scalability be used to ensure that token services are readily available to respond to requests.

#### Access Control Approaches

Access control revolves around security policy. If access control policy can be discovered and processed, and if authorization credentials of actors can be retrieved, access control can be successfully enforced. Architectural flexibility for authorization is achieved by logically separating duties into Policy Decision Points (PDPs) and Policy Enforcement Points (PEPs). A PDP is the point at which access control decisions are made, based on an expressed access control policy and an actor’s authorization credentials. The enforcement of the decision is delegated to a PEP. Some standards, such as XACML (the eXtensible Access Control Markup Language), decompose the policy model further into Policy Administration Points (PAPs) that create policy and the Policy Information Points (PIPs) that query attributes for actors requesting access to resources. There are many strategies for how PDPs and PEPs can work together, each with architectural implications that have an impact on security, performance, and scalability.

As access control policy may vary between ownership domains, the negotiation of access control policies between such domains must occur at the start of the relationship, regardless of the underlying architectural approaches.

Different security services implementations may dictate different architectural approaches and have different implications. This section provides a brief overview of such approaches.

##### Centralized Access Control Approaches

A centralized approach uses a policy server (or a set of policy servers) to act as a PDP, and utilizes the current access control policy to make an access control decision for an actor requesting access to a resource. A positive aspect of this approach can be information hiding because services may not need to know the authorization credentials of the actor or the specific policy being enforced. The centralized model protects that information in cases where this information may be sensitive or confidential. Another positive aspect of this approach is that the policy services can provide access control decisions consistently, and any change to access control policy can be changed in one place.

However, negative aspects of this model are those common with any type of centralized architecture, including performance and availability. Given performance, availability, and scalability concerns, any centralized solution should be coupled with alternative approaches for greater flexibility.

##### Decentralized Access Control Approaches

In a decentralized approach, the service consumer propagates a token related to its identity (and possibly other identities in a service chain), and this is assessed by a “local” PDP and PEP. The service PDP refers to locally expressed policy, and therefore, its PDP can inspect the policy and the security credentials propagated in order to make an access control decision. If only identity information about the initiator is propagated into the service, the service may retrieve additional authorization credentials from an Attribute Service lookup based on the identity.

The decentralized model alleviates the performance concerns of the purely central model, as it does not require access to a set of centralized servers used to make access control decisions. Because the policy is locally expressed, the service may enforce its own policy, expressed in its service contract with service consumers.

There are two potential concerns with this model. One concern is that there is no information hiding. If an assertion about the initiator is propagated into the service, the service may need security credentials of the consumer in order to execute access control policy, and these credentials may be sensitive or confidential. A second concern revolves around access control policy management. As this decentralized model is based on making “local” (not centralized) access control decisions at the service level, there is a possibility that

* Access control policies may not be consistently enforced throughout the SOA ecosystem
* Changing organizational access control policies require policy changes throughout the SOA ecosystem (vs. in a central location) and may be therefore difficult to immediately enforce. Therefore, there is a danger that access control policies may be out-of-date and inconsistent.

It is therefore prudent that in using such an approach, that these concerns be addressed.

##### Hybrid Access Control Approaches

A purely centralized approach has significant weaknesses related to performance, availability, and scalability; a purely decentralized approach does not support a requirement to have centralized control of access control policy. In response, hybrid approaches have emerged to provide a “happy medium”. between local control of policy (where services express all policy) and central control of policy (where a central policy server expresses all policy). In hybrid models, each service can both express local policy and leverage global organizational policy (which can be periodically downloaded or syndicated to the local services) in order to make decisions. The balance between the models will depend on the context in which the hybrid is applied.

### Architectural Implications of SOA Security

Providing SOA security in an ecosystem of governed services has the following implications on the policy support and the distributed nature of mechanisms used to assure SOA security:

* Security expressed through security messaging policies **SHOULD** follow the same architectural implications as described in Section 4.4.3 for policies and [contracts](#Contract) architectural implications.
* Security policies **MUST** have mechanisms to support security description administration, storage, and distribution.
* Service descriptions **SHOULD** include a sufficiently rich meta-structure to unambiguously indicate which security policies are required and where policy options are possible:
* The mechanisms that make up the execution context in secure SOA-based systems **SHOULD**:
  + provide protection of the confidentiality and integrity of message exchanges;
  + be distributed so as to provide available policy-based identification, authentication, and authorization;
  + ensure service availability to consumers;
  + be able to scale to support security for a growing ecosystem of services;
  + be able to support security between different communication means or channels;
* Common security services **SHOULD** include the ability for:
  + authentication and establishing/validating credentials
  + retrieval of authorization credentials (attribute services);
  + enforcing access control policies
  + intrusion detection and prevention;
  + auditing and logging interactions and security violations;

1. In practice, there are perceptions of security from all participants regardless of ownership boundaries. Satisfying security policy often requires asserting sensitive information about the message initiator. The perceptions of this participant about information privacy may be more important than actual security enforcement within the SOA ecosystem for this stakeholder. [↑](#footnote-ref-1)
2. Such risks and others are documented in *K. Smith, "Mitigating Risks Associated with Transitive Trust in Service Based Identity Propagation", Information Security Journal: A Global Perspective, 21:2, 71-78, April 2012* [↑](#footnote-ref-2)