Inter-enterprise Contract Architecture For Open Distributed Systems: Security Requirements

Zoran Milosevic, David Arnold, Luke O'Connor, {zoran, arnold, luke}@dstc.edu.au
Distributed Systems Technology Centre, The University of Queensland QLD 4072, Australia

Abstract

An important element of electronic inter-enterprise interactions is support for a rapid and cost-effective establishment of contracts, monitoring of parties' performance to the contract and possible enforcement capability. We have previously developed a business contract architecture to address these requirements [6,7]. This paper presents an extension of the earlier work. The concept of binding is used to describe the interactions between the components of the contract architecture so that standard contract sequences can be stored in a public repository for future reuse. Various security requirements of the architecture are identified, from the application and distributed infrastructure perspectives. It is then shown how a secure contract architecture can be derived to address these requirements.

1 Introduction

The joint potential of open distributed systems (ODSs) and the Internet is an important factor to further promote all aspects of electronic commerce (EC), including inter-enterprise business dealings. The capabilities of ODSs can be exploited to *i)* express semantics of common contract documents and enable their storage in a publicly accessible repository, *ii)* describe different roles involved in performing contractual operations and their authority, permission and responsibilities and, *iii)* model typical contractual scenarios as reflected in the interactions between trading partners and their various temporal orderings. The Internet enables cost-effective information exchange between enterprises at a global level.

In spite of the promises of both technologies, a supporting architecture is still lacking to address an important aspect of electronic business activities conducted globally. A legally sound framework for typical operations associated with establishing *contracts* and monitoring performance of the parties to the contract is required. Unfortunately, current EDI technologies appear to be inadequate and too expensive to provide such support.

Our work on developing a business contract architecture can be seen as a step towards achieving such support. Over recent months we have indeed observed several commercial products which address some of our concerns raised in [6,7] with several examples offered in [13, 14].

These products offer useful though partial solutions to various problems associated with rapid, flexible and *secure* support for inter-enterprise EC. In particular, the notion of security includes many facets, ranging from security requirements of applications, via the security of distributed middleware technologies through to security of underlying communication and operating system technologies. While some progress has recently been made in providing more secure EC transactions over the Internet, (e.g. secure payments), substantial work is still needed to provide a sound *security architecture* that addresses a full set of requirements for global EC activities.

In this paper we investigate security requirements pertinent to inter-enterprise interactions by using a *business contract architecture* (BCA) previously developed [6,7]. We use the BCA because it is a technology-independent architecture, with a rich set of security requirements from the application, and distributed infrastructure perspectives.

Section 2 outlines the BCA and proposes the use of workflow to automate contractual procedures. Section 3 describes implementation of the BCA using two distributed environments. Section 4 addresses architectural issues mentioned in the previous two sections from different security perspectives. Section 5 discusses security related work which we plan to undertake in near future.

2 Basic concepts of the BCA

Based on observations of how contracts are viewed from economic, legal and business points of view, we have developed a BCA which reflects typical operations associated with contract negotiation, validation, monitoring and enforcement [6,7]. The basic architectural components of the BCA are as follows.

The *Contract Repository* (CR) provides a common understanding of contract types. It stores:.

- Some general *contract element types* which are common to many contracts (e.g. the date of the contract agreement, duration of the contract, parties to the contracts, etc.).
- Different *contract types* ranging from very simple contracts (no long-term relationships, the items traded are well defined); to very complex contracts, such as those governing international inter-organisational contracts.

- *Common contractual scenarios*. Since many contractual activities in the real world involve scenarios that are to a great extent common across organisations, these scenarios can be stored in a public repository to facilitate future *reuse*.

Descriptions of these types are introduced to the contract repository through a special trusted authority called a *contract administrator.*

The *Notary* stores contract instances after the contract has been agreed upon and checked for validity. This can be later used as evidence of agreement in the contract monitoring and enforcement activities.

The *Legal Rules Repository* (LRR) stores the rules and policies of a particular legislative domain.

The *Contract Validator* (CV) ensures the creation of legally valid contract instances. This includes the checking of the following aspects of contract validity.

- The *competence* aspect. To accomplish this, the CV verifies the capacity of parties willing to enter a contractual relationship (see 4.1 below).
- The *clarity* aspect of a contract template. In most cases the contract semantics will be unambiguous if it is derived from a template in the CR. The CV can be used to provide additional checking should a need for this arise.
- The *legal purpose* element of a contract, based on the information in the legal rules repository. This is done through the *contract legality* (CL) object.
- The *consideration* element of contract. This can be done by checking whether the contract template contains those contract elements which describe what is exchanged between the parties.

The *Contract Negotiator* (CN) mediates the negotiation process (alternatively, this can be carried out by the parties themselves). During this stage the parties can exchange several contract templates (offers and counter-offers) and the contract template may be submitted for validity checking.

The *Contract Monitor* (CM) enables monitoring of the activities of parties, measuring their performance and recording the relevant events. It can also signal a contract non-performance to the contract enforcer if it detects such an event.

The *Contract Enforcer* (CE) makes an enforcing action, upon being signalled by the CM, either *i)* directly on the parties to ensure that the actual behaviour conforms to the contract, or *ii)* by informing the CV which may prevent further access to the system by non-conforming parties. While the pro-active approach to contract enforcement is not widely used in business contracts, we envisage that this can be done in an EC system. A business law is normally less ambiguous than common law and direct interpretation is possible. Reactive and post-contract enforcement will usually require arbitration and possibly human intervention in determining the appropriate (corrective or punitive) actions.

A contract 'life-time' includes *i)* contract establishment, which includes contract negotiation and validation procedures and *ii)* contract performance, which is related to the behaviour (performance) of parties to the contract and may include monitoring and enforcement activities. In many cases the order of these steps can be automated and thus it can be anticipated that *workflow management systems* will be well suited for some of these sequencing.

Since workflow systems are at present designed for the use within organisational boundaries, some necessary adaptions are need to facilitate workflow execution in an inter-enterprise setting. This raises several novel security requirements. For example, contract negotiation involves several activities which need to be performed in order to ensure authorised initiation and signing of a contract offer which will be sent to another trading partner.The sequence of the contract steps is given in more detail in section 4.1 to include some relevant security aspects.

3 Realisation in distributed environments

The BCA can be realised by using different middleware technologies. We indicate how this can be done by using a CORBA complaint platform [10], and the DSTC Architecture model [2] (referred to as 'DSTC-A' hereafter).

The CORBA Interface Repository (IR) represents an elementary type repository. It can used as a persistent storage which contains the CR with the contract types. In fact, an IDL file which contains the description of different contract types can be compiled and stored within the IR by an authorised administrative body, responsible for creating and administering contract templates. These can then be retrieved at runtime, by both servers and clients, using IR supplied functions.

The fundamental contract operations can be implemented in different ways. In order to accomplish a sufficiently generic solution, a *contract interface* can be created which includes negotiation, monitoring and enforcing operations. Such a contract interface can then be used by any server that requires contractual operations: the server's interface inherits properties of the contract interface.

Further, each of the BCA components, can be implemented as the corresponding object which interface specifies the operations associated with these roles[1].

The CORBA environment however, is inadequate to store the templates for the interactions between objects involved in providing a service. In other words, we can store the representation of common contract *documents* which reflect semantic aspects of contracts, but it is not

[1]. A detailed description of a prototype implementation using CORBA is beyond the scope of this paper and can be found in [6].

easy to store the semantics of contract *scenarios* as also required by the BCA. To address this problem we propose the adoption of the *binding* concept introduced within the RM-ODP [3] and further refined within the DSTC-A [2].

A binding is an association between a set of objects that allows objects to interact. Bindings are strongly typed - a *binding type* defines the roles of the objects in a binding and the interaction that can occur between objects fulfilling those roles [2]. Each role of a binding specifies an *interface type* that must be satisfied by objects fulfilling that role. A binding includes multiple participants, a definition of the interaction between them and a specification of obligations and requirements of each of the participants and of the environment itself. Binding semantics thus provides a template for interactions between objects.

We contend that the binding is also a suitable means to model enterprise related interactions, such as those associated with business contracts. Everyday experience suggests that there is a large (but still finite) number of scenarios associated with business contract interactions. These for example can be related to different *i)* temporal ordering of interactions, *ii)* dependencies between the entities to the contract, *iii)* dynamically changing number of participants involved during the contract life time (both the parties to the contract and the BCA components) and *iv)* relationships between contract domains.

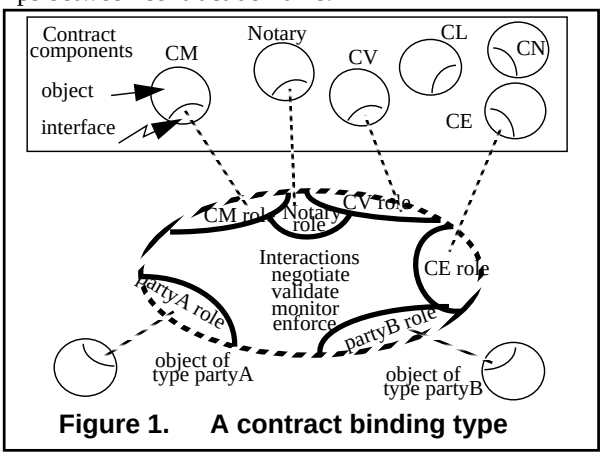

Figure 1. A contract binding type

These different contractual scenarios, expressed in terms of a workflow process model, can be described through the corresponding binding types. An example of such a binding type, is depicted in Fig. 1. This binding type includes *i)* the roles of *CV, Notary, CM* and *CE* (but not *CN* or *CL*) and the roles of two parties to the contract, *partyA* and *partyB* (in general a binding can specify roles of many parties to the contract) and *ii)* supports interactions which embody contract validation, notary operations, contract monitoring and contract enforcement.

We anticipate that there will be various relationships between binding types which embody semantics of the corresponding contractual arrangements. For example, the *subtyping* relationships can be used to form an appropriate 'contract binding type hierarchy', in which a base 'contract binding type' can be a subtype derived from the binding type.

In addition, the standard contract interactions associated with contract life-cycle can be used to construct required contract binding types. For example, a 'contract binding negotiation type' can describe basic negotiation related interactions associated with contract templates i.e., offer, counter-offer, acceptance and rejection. In the case of a successful negotiation, this binding type specifies that these interactions are recorded by a notary object for use as evidence of agreement in the contract validation and enforcement activities.

While we recognise the potential of the binding type to model enterprise interactions, we note that the binding types introduces new *security* requirements as will be discussed in 4.2.

4 Security requirements for the BCA

The concepts related to the BCA bring many new security requirements. These need to be met in the context of *i)* the implementation of the behaviour of the BCA components and their relationships, *ii)* a workflow oriented description of the sequences of contractual operations, including their inter-organisational aspects, and *iii)* the realisation of the workflow through the use of the binding concept.

This paper focuses on tow perspectives to ensuring secure BCA operations. The *application* perspective is concerned with specific components needed to address enterprise security concerns. An example is support for verifying the authenticity and authorisation of people assigned to specific roles within a company to permit them to perform contractual operations. The *distributed infrastructure* perspective is concerned with enforcing security policies at the level of objects which perform functionality associated with contracts and the bindings which control their interactions.

4.1 Application perspective

There are three application specific security requirements of the BCA needed to ensure its secure operations, as elaborated below.

First, each of the BCA components need to be able to be *authenticated*. We adopt the public-key cryptography [4] so that any party wishing to use a BCA component can exploit this security mechanism. Each party has a certificate (which binds its name and other identifying information to an encryption key) and a pair of encryption keys - a public key and a private key, with the public key being the key bound to the certificate. Certificates are registered with a Certificate Server, returning any certificate requested by any party.

The public key derives its name from the fact that it is bound to the certificate and the certificate is publicly available from the Certificate Server. Each party keeps its corresponding private key secret.

Such a security infrastructure permits messages to be sent privately, and to be digitally signed (which in our context is a contract or part thereof). The validity of the signature can be checked by retrieving a set of public keys from an appropriate set of certificates. We illustrate the use of signatures by annotating our original proposal for the sequence of operations for generating a contract between parties A and B [6].

When party A retrieve a contract template C from the CR server, a signature is included by the CR which testifies to the correctness of the template. Party A can verify the signature by retrieving the public key of the CR through its certificate. Once the competency of the contract has been established (discussed further below), A and B exchange the template, assigning values to the elements of the C until both agree on the final contract. At each step of the negotiation each party signs its current offer, and if privacy is an issue, also encrypts their reply. At the end of the negotiation both parties have a copy of the contract signed by the other party, which is taken as legal proof of the contract's existence. If the contract was negotiated through a third party then the third party may also sign the contract.

After the contract has been negotiated, both parties check its legal purpose. This is performed, by consulting a LRR, accessed through the CL server. Again, the responses from the CL will be signed to verify their authenticity. If both parties are satisfied then the contract instance is stored in the Notary using the public key system as previously. The CM can now start monitoring activities of the parties if required. Both parties are bound to the contract by virtue of their signatures on the final contract which can be checked by the CM. If the contract is violated the CM can send signed requests to the CE server to take action over the violation. If any dispute arises, the sequence of signatures derived from the contract negotiation and signed information from the CM and CE can be used for arbitration.

Second, parties willing to use the BCA components need to establish a high level of *trust* in these components. To this end we adopt the security mechanisms proposed in [5]. These are based on the 'real life' notions of endorsements, licences, insurance policies and surety bonds. These assurances can be represented as electronic documents, certificates digitally signed by the respective bodies, viz endorsers, licensing authorities or insurance providers. With this security mechanism in place, the parties willing to use BCA can now place more confidence in all BCA components. They could accept assurances from trusted agencies such as licensing, endorsements or insurance agencies about the security and competence of these components.

Third, in order to ensure the validity of contracts, the *competence* element needs to be satisfied. In the context of the BCA this means ensuring the competency of the parties to the contract. This is essentially the problem of determining if a given person has the authority to establish a contract. We assume that the set of possible contract templates C_1, C_2, ..., C_{CR} are well-known and can be obtained from the CR as signed documents. It is the responsibility of each company T to peruse the set of contracts and to determine which contracts it will enter into, and who may negotiate the contracts. To this end, each company will establish a publicly accessible Competency Server that contains a list of contracts accepted by the company and a set of procedures to be used to verify the competence of each contract offered.

Our proposal is to base competence on a notion of *roles*, which reflect the structure of a company, e.g. CEO, presidents, state managers, managers, and administrators. As is common practice, a person can proceed with a request if the permission is obtained from some collections of the superiors such as 3 managers or 2 state managers, or simply the permission of the CEO. We use digital signatures and roles to implement competency that can be verified by people within a company and by those negotiating with the company.

For a given company T let the set of possible roles be $R_T = \{R_1, R_2, ..., R_n\}$, where we may have R_1 = CEO, R_2 = state manager and so on. We call R_T the role profile of company T. Now, if C_i is a contract that company T wishes to offer, and R_j is the role of a party that can negotiate C_i, then T defines a set $S_{\{i,j\}} = \{S_1, S_2, ..., S_m\}$ where $S \subset R_T$, is called a competency set. The meaning of the set $S_{\{i,j\}}$ is that R_j is deemed competent to negotiate contract C_i if R_j can gain the digital signatures of parties that represent all the roles in at least one of the $S_k \subset S_{\{i,j\}}$. The sets $S_{\{i,j\}}$ are known as the Competency Profile of the company. Competency Profile and in particular the sets S_k reflect security policies of organisations regarding the approval processes (these policies can be quite complex). Both the role profile and the competency profile are digitally signed by a Competency Server that monitors changes to the profile.

As an example, consider party A attempting to negotiate contract C_i with party B. Party A initiates a self-competence check by retrieving its company's Role Profile and Competency Profile. The public keys of the Role Server and Competency Server are retrieved and the signatures on the respective profiles are verified. Party A then examines the entry $S_{\{i,A\}}$ in the Competency Profile

and recovers the competency sets S_1, S_2, ..., S_d associated with their role in negotiating C_i. A then selects a subset of parties P in the company such that for at least one S_j we have that $S_j \subset P$, and requests that each person in P sign a message to the effect that they approve of A negotiating contract C_i. These requests may be done via email or using a workflow model, for example. Once at least one competency set S_j has been covered by the signatures of responding parties, A retrieves the certificates for the parties representing the roles specified in S_j and then verifies the signatures. We assume that when a certificate is created for a party their role is included in a role field. Party B can follow the same procedure to perform a self-competency check, and further, since the verification of competency only relies upon public information (the role profile, the competency profile and certificates) then A and B can check the competency of each other.

This approach addresses some of the security issues related to the use of a workflow to control the sequence of contractual operations in an inter-enterprise context. This is also an important future research topic.

4.2 Distributed infrastructure perspective

The BCA can also be realised by using the concept of a distributed architecture such as DSTC-A model. In this case, the BCA components can be implemented as objects whose interfaces abstract their behaviour, and further the interactions between them can be implemented via the binding. The underlying distributed infrastructure provides necessary functionality for the dynamic instantiation of interfaces and objects, creation of bindings and addition/ removal of objects to binding. All this functionality needs to be implemented in a secure fashion. To this end the infrastructure should also provide necessary *security services*. These are required for secure functioning of the BCA but also for many other services in an ODS. The security services and the BCA requirements for these are listed as follows[2].

- The *integrity* and *confidentiality* of objects and their communication. The integrity means the protection of information against the threat of modification by unauthorised users while confidentiality is protection against disclosure to unauthorised users. Both of these are important for the interactions between the objects which realise BCA components and for parties to the contract.
- *Authentication* of objects, the trusted process of validating an identification claimed by a principal (an object or a human) of the system. This process must prohibit other principals from masquerading as other principals [8]. An object can have several interfaces which abstract different object's behaviour (e.g. provision of a computational service, or a management functionality). This suggests that an object can have different identities reflecting different aspects of its behaviour such as its real identity (used for initial authentication), operational identity (e.g. to be included in access control lists), and charging identity (whom to charge for the use of resources) as also noted in [12]. Each of the BCA objects need to be authenticated and also the objects representing parties to the contract, after their competence have been previously verified. The latter can be regarded as the authorisation at the application level.
- *Access control* (or authorisation) - for enforcing access security policies whose rules state under which conditions each of the objects can accept a binding proposal from other object(s) on the basis of their identity**.** The novel requirement arising from the binding is ensuring multiple *secure channels* between objects interacting in a binding. It is these channels along with the controlling security objects that should provide a basis for trusted interactions between components. There are many different access policies relevant to the BCA objects. For example, an object representing a party to the contract must not have an access to the CM or CE objects, but these objects will need to have access the objects representing the parties (for monitoring their performance and also some performative actions such as sending warnings or even breaking bindings between the parties if one of the parties misbehaves).
- *Delegation* of rights from one object to another. It is often the case in ODSs that the objects providing a service require the services of other objects. Consequently, a complex graph of interactions can result whereby objects invoke operations on other objects' interfaces (or even create new object instances). This requires a means to pass on security information such as access control among objects and specify constraints which govern this. For example, parties involved in a potential contract can delegate some of their rights to the CN for performing negotiation operations on their behalf.
- *Auditing* of all object activities. This includes recording of the activities of the objects in the system with the aim of detecting actual or attempted security violations. This is required for monitoring the proper functioning of all the BCA components and in particular for monitoring activities of the parties to the contract which is the functionality of the CM object.
- Support for preventing *denial of service* type of attacks which can result from events such as other legitimate users making abusive use of network or distributed sys-

[2]. These are developed to meet the requirements of our distributed infrastructure, in particular those associated with binding. Many of the issues are also pertinent to recent CORBA security work [11]

tem resources. In the context of the BCA this is required by the CE object.

- Security *management* capabilities. This includes defining the different security policies and security domains within which they are valid, and administering them. The use of the domains enables dealing with scalability and efficiency issues.

These security services should enforce *security policies* (defined as a set of rules that constraints one or more sets of security relevant activities of one or more sets of objects [8]). These security services should be positioned within the infrastructure of the DSTC-A model. Some of these can rely on standard security *mechanisms*, such as public key and secret key encryption mechanisms and Generic Security Services API (GSS-API) to allow flexible use of encryption services [15].

Many of security services however need to be developed to address the specifics of the distributed infrastructure. This is an area which requires substantial future work. In the course of the development of an infrastructure to support DSTC-A model, we have implemented some security support. First, in the implementation of our architecture model, objects provide a mediation method used to evaluate binding proposals. This method is free to use all and any services available to it to make its decision on the proposal. Typically, this would be authentication and authorisation checks on the proposed participants. Second, we also started to address the problem of supporting security sessions (contexts) between more than two parties. As stated previously, this represents an important requirements for security in the context of binding. A design and an initial implementation for secure multiparty sessions [1] has recently been completed and is now in the process of being tested. Additionally, support for defining and administering policies (in generic sense) is currently nearing completion and this will represent a good base for building support for security policies.

5 Open issues

While some current middleware technologies provide certain aspects of security mechanisms (e.g. DCE adopting Kerberos system [9]) it is recognised that what is needed is a *security architecture* which would encompass all the security issues many of which were mentioned in this paper. For example, work is currently carried out to provide such an architecture for the CORBA compliant platforms [11]. Additionally, new impetus towards a broad range of EC activities promoted by the Internet brings new security requirements, such as those related to secure payments over an open network.

In our future work we plan to address several open issues related to a secure ODSs in general and the BCA in particular. This includes:

- interoperability between different security mechanisms
- revocation of the authorisation, i.e. how can privileges be revoked in a widely distributed system
- expression of security policies
- scalability of authentication schemes and authorisations
- realising sequences of contract operations using the binding concept and including useful workflow controls within it.

Acknowledgments

The work reported in this paper has been funded in part by the Cooperative Research Centres Program through the department of the Prime Minister and Cabinet of the Commonwealth Government of Australia.

References

[1] D.P. Barton, L.J. O'Connor. A Design and Implementation of Secure Multiparty Sessions, DSTC Internal report., Jan. 1996.

[2] A. Berry, K. Raymond. The *A1✓* Architecture Model. In, ed., K.Raymond, L.Armstrong, Open Distributed Processing; Experiences with distributed environments, Proc. of the 3rd IFIP TC6/WG 6.1 International Conference on Open Distributed Processing, pages 55-67, Chapman & Hall, 1995.

[3] ISO/IEC IS 10746-3. International Standard 10746-3, ITU-T Recommendation X.903: Open Distributed Processing: Reference Model,,Part 3: Architecture, 1995.

[4] ISO/IEC 9594-8: Information Technology - Open Systems Interconnection - The Directory: Authentication Framework (Also ITU-T X.509).

[5] C. Lai, G. Medvinsky, B.C Neuman. Endorsements, Licensing, and Insurance for Distributed System Services. In Proceedings of the 2nd ACM Conference on Computer and Communications Security, pages. 170-175, Nov. 1994.

[6] Z. Milosevic. Enterprise Aspects of Open Distributed Systems, A PhD thesis, Computer Science Dept., The University of Queensland, Oct.1995.

[7] Z. Milosevic, A. Bond. Electronic Commerce on the Internet: What is Still Missing? Proc. of the 5th Conf. of the Internet Society, p. 245-254, Honolulu, June 1995.

[8] G. Mohay. The DSTC Security Model A1-SM. DSTC Internal report, May 1994.

[9] B.C. Neuman, Theodore Ts'o. Kerberos: An authentication Service for Computer Networks, IEEE Communications, Sept. 1994, pp 33-39.

[10] Object Management Group. The Common Object Request Broker: Architecture and Specification, Revision 2.0, 1995.

[11] Object Management Group. CORBA Security, OMG Document Number 95-12-1, December 1995.

[12] Sesame Consortium. Sesame V4 - Overview, Issue 1, December 1995.

[13] Certification for Electronic Commerce, accessible at: http://www.batnet.com/cec/

[14] Digital Notary Service, accessible at: http://www.surety.com/about-dns.html

[15] Generic Security Service Application Program Interface, Internet RFC 1508.