Supporting Accountable Business to Business Document Exchange in the Cloud

Derek Mortimer
School of Computing Science
Claremont Tower, Newcastle University
Newcastle upon Tyne, England
Email: d.j.mortimer@ncl.ac.uk

Nick Cook
School of Computing Science
Claremont Tower, Newcastle University
Newcastle upon Tyne, England
Email: nick.cook@ncl.ac.uk

***Abstract*—Business-to-Business (B2B) interactions can be defined in terms of the exchange of documents. Such exchanges must be regulated to comply with obligations including service agreements, contracts and law. It is fundamental to such regulation that participants are held accountable for their actions. This can be achieved through the deployment of support services to produce a verifiable audit trail of interactions. Typically these support services have been designed to be deployed in-house, placing requirements on infrastructure and technical ability. Cloud and utility computing allow this support to be delivered as a cloud hosted service by a specialist security provider, alleviating the aforementioned concerns for businesses. This paper analyses various possibilities for leveraging the cloud in this way and reports on the design and implementation of a document exchange service constructed using cloud-based infrastructure.**

Keywords-cloud computing; service oriented computing; business-to-business; accountability; non-repudiation;

I. Introduction

The ubiquity of services and resources available across the Web presents increasing opportunities for business-to-business (B2B) collaboration between organisations to achieve mutually beneficial goals. These collaborations are conducted in the form of exchanges of business documents. Participants in such collaborations maintain their autonomy except with respect to their agreed undertakings but will typically privilege their own interests above those of their partners. This preference can lead to a tension with the desire to cooperate. Recent studies show a rise in both network-based collaboration (or "the extended enterprise") and the inherent risks arising from increased collaboration [1]. The study shows that data breaches involving business partners had risen from 8% in 2004 to 32% in 2008, peaking at 44% in 2007.

To mitigate risks associated with collaboration, there is a need for monitoring and control of B2B interactions, allowing behaviour to be regulated to ensure compliance with agreements such as service contracts and process specifications.

An important aspect of monitoring and control is accountability. Partners must be held accountable for their actions and all actions taken must be acknowledged. To this end, business partners agree to the generation of an audit trail that provides evidence for accountability. Without this unambiguous evidence agreements can be unenforceable and subsequent dispute resolution made difficult. For example, it should not be possible to subsequently deny having received or transmitted some business document in the context of a regulated partnership [2].

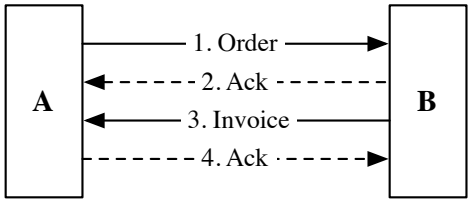
Generation of an audit trail can be achieved by deploying support services at the middleware level, these support services operate as lower level mechanisms in B2B interactions, functioning to assist in realising higher level views of B2B collaborations, such as those described in [3]. By deploying such support at the middleware level, businesses are freed from underlying technical concerns, allowing them to focus on meeting business objectives.

Support services fulfilling this functionality have been discussed in previous work [4], [5], however these services were designed to be hosted inside an organisation wishing to conduct fair exchanges with other parties. This placed requirements on expertise and infrastructure that many small to medium enterprises are unable to satisfy. Cloud and utility computing have the potential to allow such capabilities to be deployed as services in the cloud. This allows expertise requirements to be shifted to a security service provider that uses the cloud to address infrastructure requirements.

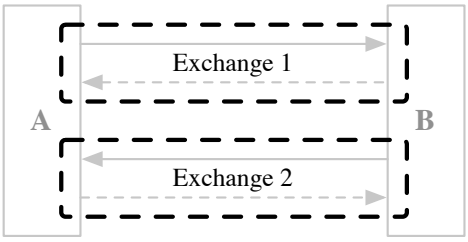
This paper describes the design and implementation of a cloud-based centralised business document exchange service. Section II will discuss the motivation for regulated exchange. Section III will introduce fair and non-repudiable document exchange. Section IV will discuss motivation and assumptions of operating the service in the cloud. Section V will explore design possibilities and implications for the cloud-based accountability service. Section VI will describe the design and construction of the centralised document exchange service and its use. Section VII will report on related work. Section VIII will discuss conclusions and future work.

II. Motivation for Regulated Exchange

This section will motivate the need for fair exchange of business documents between partners for regulation. Businesses interact by exchanging business documents to achieve mutually beneficial goals. These documents typically include



(a) Multiple messages comprise a business conversation.



(b) The same conversation can be broken down into constituent exchanges.

Figure 1. A business conversation shown fully and broken down into a sequence of exchanges.

purchase orders, requests to tender or contracts such as non-disclosure agreements. They may have intrinsic value or be critical to the success a partnership.

Document based exchanges, naturally modelling these real world B2B interactions, have led to the development of open standards such as ebXML [6] and RosettaNet Partner Interface Process [7] that specify business conversations. These conversations detail the structure, ordering and basic requirements (e.g. deadlines, acknowledgements and reliable delivery) of the messages to be exchanged to successfully achieve some business objective. The payload of these messages are the business documents and acknowledgements being exchanged.

Figure 1 (a) shows an example conversation including the delivery of an order and the production of a corresponding invoice, both with acknowledgements. Information such as deadlines for transmission and processing are omitted for brevity.

While these conversations may involve many steps, they can be broken down into correlated sequences of single exchanges, each with optional acknowledgement as shown in Figure 1 (b). Conversation identifiers within each message allow correlation of exchanges.

The regulatory challenge is to ensure both participants comply with the required behaviour. Specifically, assurance is needed that for all single document exchanges from A to B:

1) There is irrefutable evidence the document originated with A.
2) There is irrefutable evidence the same document was received by B.
3) Neither A or B obtains documents or evidence prematurely.

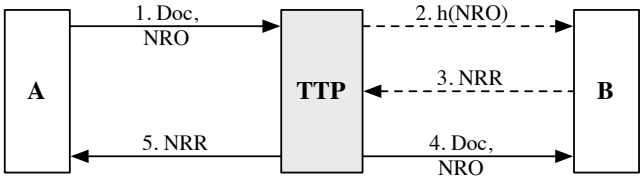


Figure 2. Non-repudiable Document Exchange.

Points 1 and 2 in the above list can be addressed by non-repudiation, the property that no participant can subsequently deny their participation in an activity [8]. Point 3 is addressed by fairness, informally defined as ensuring that no well behaved party is placed at a disadvantage through the misbehaviour of another [9]. Failing to address these points can place participants at a disadvantage despite correct behaviour. In contrast, the outcome of a fair, non-repudiable exchange is that all parties obtain the evidence and documents to which they are entitled, or the ability to acquire them, or that none do.

Non-repudiation is recognised as an important property of standardised business conversations. Current standards allow conversations to be specified as requiring non-repudiation but do not prescribe the mechanisms by which this must be achieved, or whether it must be done fairly.

III. Non-Repudiable Document Exchange

The previous section shows accountability is essential for regulating B2B interactions and introduced the notions of non-repudiation and fairness. Previous work demonstrates deterministic fair exchange requires a trusted third party (TTP) [10]. The assumption adopted for this paper is that participants trust their exchange service but not necessarily each other. Thus, non-repudiable document exchange is the fair exchange of a document and irrefutable evidence of its origin for irrefutable evidence of its receipt, aided by a TTP (i.e. the exchange service) as shown in Figure 2.

The document exchange illustrated in Figure 2 is derived from the Coffey-Saidha non-repudiation protocol that uses an in-line TTP for communication [11]. Communication channels between A and the TTP, and between B and the TTP, are encrypted.

The NRO token shown in the figure represents *Non-repudiation of Origin*, specifically A's digital signature over Doc. The NRR token refers to *Non-repudiation of Receipt* and is B's signature over the NRO evidence, creating a cryptographically bound chain of evidence from Doc through its submission by A and its retrieval by B. Specifically for Coffey-Saidha non-repudiation, NRR is generated by B signing a digest of NRO, to prevent B prematurely obtaining NRO.

Execution of steps 1 through 5 as shown in Figure 2 represent one complete execution of the Coffey-Saidha protocol.

For the case of document exchange this can be broken down into three phases.

Step 1 constitutes the *submission* phase, in which A submits both Doc and NRO to the TTP. Steps 2 through 4 constitute the *retrieval* phase in which B receives a digest of A's NRO evidence, digitally signs it (generating the NRR evidence) and submits it to the TTP, subsequently gaining access to Doc and the original NRO. Step 5 is the final *evidence retrieval* phase in which the NRR evidence is made available to A for collection. The phases must be executed in the correct order to maintain fairness.

For brevity, full non-repudiation protocol details have been omitted including required cryptographic information, protocol run identifiers and trusted timestamps, see [4] for detailed discussion on the topic.

The protocol provides deterministic fairness by using the TTP in every step of an exchange between two participants (*inline*). Other approaches relax TTP involvement by using it only for initiation and guaranteed termination (*online*) or by using it only for guaranteed termination (*offline*). Zhou [12] and Kremer et al [8] provide surveys of the various approaches and their properties. The exchange service described in this paper will act as a TTP to support non-repudiation protocol execution on behalf of its customers.

Previously, participants such as those in Figure 2 are required to understand the lower level non-repudiation protocols and their associated evidence to be able to partake in an exchange. A motivation for providing accountability as a service is that the participants can be freed from many underlying technicalities to focus on business objectives.

The exchange service described in this paper renders accountable document exchange as an explicit service. Future work could provide accountability as a satisfiable property of message delivery middleware, further shielding the participants from underlying details. This will be discussed in section VIII-A.

IV. Motivation and Assumptions for Cloud-based Operation

Section I introduced previous work that focussed on hosting support services for accountability within the individual entities wishing to engage in non-repudiable exchange of messages or documents [4], [5]. Hosting these services within a business requires both infrastructure to execute the services and technical knowledge to configure them correctly.

In discussing the motivation for designing and operating the service in the cloud, two target groups are discussed. The first group is that of the security service provider. The security service provider has the technical knowledge to design and operate an accountable exchange service, including understanding the related socio-legal implications. However, they do not not necessarily have the required infrastructure to support it.

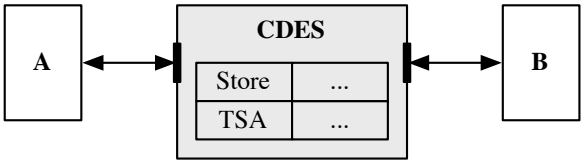


Figure 3. Exchange through a centralised service.

The second group is the service consumer, presumably characterised by small to medium enterprises (SMEs) but more broadly described as any business wishing to engage in accountable exchange of documents but without the technical know-how or the infrastructure to provide the required support services themselves.

By operating the service in the cloud the service provider is allowed to capitalise on their expert knowledge, spending only what is required on the cloud provider for the resources consumed. The service consumer is able to engage in accountable exchange by simply consuming the accountability service, presumably paying only for their slice of the consumed resources directly to the service provider who in turn pays the cloud provider(s).

A secondary benefit of cloud-based operation is that resources are assumed to be easy to acquire. This allows the possibility for the exchange service to scale to support large numbers of consumers, documents store and concurrent exchanges. Details of such scaling will be discussed in section VI.

The assumption in section III is that consumers trust the exchange service but not necessarily each other, this leads to the assumption that the service provider must trust the cloud providers on which their service is executing. The possible use of trusted computing to increase confidence in the security guarantees of cloud providers is discussed in section VII.

V. Accountability as a Service

As previously discussed, cloud technologies can be leveraged to provide accountability as a service and make this service operate at a cost of only what resources (e.g. storage, bandwidth and computation) are consumed.

When dealing with sub B2B exchanges, companies may trust their own decisions and processes ahead of their collaborators. In the case of accountability as a service this trust extends to the business' choice of service provider, leading to two useful scenarios to consider in terms of document exchange services.

Figure 3 illustrates the centralised exchange scenario, under which a single service facilitates an entire exchange between A and B. The service provides points of interaction for both clients and accesses all necessary component services to function as a TTP for the exchanges (e.g. Trusted Timestamping Authority (TSA) and storage of documents, evidence and logs).

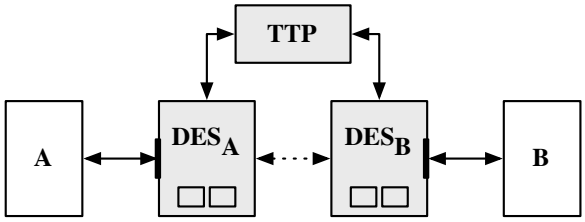


Figure 4. Exchange through decentralised services with an external TTP.

Under the centralised exchange scenario, A and B have to agree that the chosen exchange service meet both of their requirements. The natural alternative is a decentralised exchange service as shown in Figure 4.

In this case, an exchange is facilitated by separate services acting on behalf of their respective clients. Figure 4 shows distinct providers for each service and the TTP. While only the TTP is shown as external to both providers here, it is feasible other elements (e.g. storage or logging) could also be located elsewhere.

The decentralised approach allows clients to consume services satisfying their own specific requirements and decouples the choice of some components from the choice of service provider. Figure 4 illustrates a scenario in which participants could agree to use a specific TTP, or delegate the decision and agreement to their respective service providers.

This decentralised approach allows flexibility but introduces new complexities. Particularly, issues of inter-provider communication. To enable multiple exchange providers to facilitate exchanges, standards would need to be defined specifying the structure and sequence of messages to be exchanged. Such standards would need to be general enough to support multiple non-repudiation protocols and deal with agreement on which protocol to use for each exchange.

For centralised and decentralised scenarios, two approaches to service design and construction were considered. These approaches centred on building everything using cloud-based infrastructure as the building blocks (i.e. 'built using the cloud') versus building the system off the cloud and deploying different elements of the system into the cloud as needed (i.e. 'deployed on the cloud').

These possibilities will be discussed further in this section by considering the major aspects of the document exchange service:

1) The processing (or computation) components of the system
2) The communication with and within the system
3) The storage of information within the system (files, databases and logs)

A. Built Using Cloud

This approach seeks to leverage cloud-hosted technologies for all system aspects (computation, communication and storage). In the case of a document exchange service this might entail the use of databases for storage, message based communication with queues and topics and virtual machine deployment for computation.

An appealing aspect of leveraging cloud technologies is the cost and speed at which resources may be acquired, potentially quicker and cheaper than purchasing and configuring hardware within an organisation. Assuming an adequate design, the ease of acquisition of these constituent technologies means the system can scale the necessary components to cope with increased number of users, documents stored and ongoing exchanges or other service requests such as evidence retrieval. This reduced cost and increased accessibility allows the motivations discussed in section IV to be realised.

When choosing components that are services themselves, there is choice with regards to using a single provider for all aspects, or multiple distinct providers.

For providers such as Amazon, there are benefits to the single vendor approach. For example, Amazon do not charge for internal data that passes between their own services, they guarantee immediate delivery of notifications from Elastic Compute Cloud virtual machines to Simple Notification Service (SNS) topics and they provide guaranteed delivery of SNS notifications to Simple Queueing Service subscriber queues. Another possible advantage to this single provider approach is that all elements may be hosted on the same logical network, within a firewall.

An obvious issue with choosing a single provider is that flexibility regarding desired availability, reliability, durability or performance is limited to only what the chosen provider offers. By allowing diverse provider choice, providers can be judged individually based on specific requirements including those mentioned and other factors such as economic concerns.

B. Deployed On Cloud

While section V-A focussed on constructing a system from the ground up to use the cloud, this approach allows a transition to using the cloud by taking existing systems and deploying them on the cloud.

Business can encapsulate one more more component services of a system and their execution environments into virtual machines. These virtual machines could then be deployed on the cloud. Deploying configured virtual machines in this manner is a relatively simple operation, allowing a near seamless migration, useful if the primary goal is to migrate an existing setup in to the cloud. This could be used to facilitate phasing out older hardware or even coping with temporary failures. Encapsulating small elements of a service like this allows for rapid entry into capitalising on the cloud.

More complex possibilities for 'on cloud' deployment include replacing individual aspects of the system with cloud-hosted technologies such as those used in section V-A,

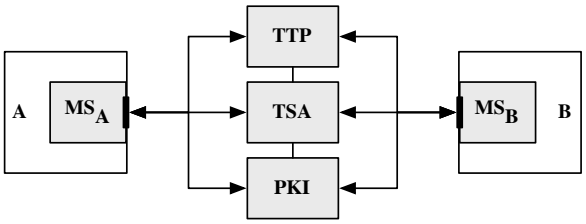


Figure 5. Conceptual components of previous work.

the distinction here is that at least one element of the system remain in the control of the customer. Reasons for maintaining this control may include an inability of cloud providers to satisfy specific requirements or laws and policies regarding data protection, Sion and Winslett [13] provide a thorough overview of regulatory issues when dealing with data management. Techniques such as anonymising data with an association identifier for reintegration after processing in the cloud can be used to deal with some of these issues.

1) Deploying Previous Non-repudiation Work On Cloud: Section I mentioned previous work involving the construction of support services to facilitate non-repudiable delivery of messages, envisioned to be executed as a messaging service (MS) within an organisation [4], [5]. Figure 5 shows the conceptual components employed in the work (for both Web Services (WS) and Java Messaging Service (JMS) technologies). The messaging service components, hosted within each organisation here were responsible for their own storage of evidence and logging.

Cloud computing allows hosting to be outsourced and it is possible that any of the components (MS, TTP, TSA, Public Key Infrastructure (PKI)) here could be quickly moved into the cloud and eventually supplanted by cloud-based infrastructure.

C. Common Issues

There are some concerns that are common to both approaches. When dealing with multiple providers as in section V-A, or even multiple components as in section V-B, the use of open standards would increase the usefulness of a design. For example, messaging standards such as AMQP [14] could render the messaging provider a simple configuration detail of the exchange service, allowing the use of both cloud and non-cloud based providers. To similar ends, Restful State Transfer (REST) [15] could be used for storage services, assuming well understood resource representations.

VI. Centralised Exchange Service

Section V introduced design possibilities for development of an exchange service and some of their associated issues. As a first example, this section describes the design and construction of a centralised document exchange service as described in section V-A and realised using Amazon Web Services.

Interaction with the service will be discussed initially, allowing internals of the service to be introduced and described in detail. The interaction with the service is message based and assumes at least once delivery for messages. Communication will be asynchronous and facilitated in a point-to-point manner by queues and in a publish-subscribe manner by topics.

Two partners, A and B, exchanging documents through the centralised document exchange service (CDES) are assumed to have prior knowledge of each other (i.e. sufficient for either to start an exchange) and to have agreed upon use of the same provider's CDES.

Fairness and non-repudiation are achieved by following the phased generation, exchange and release of a document and its associated evidence as described in section III.

A. Interaction with the Service

Figures 3 and 4 illustrate that where customers wish to interact with their exchange service, the service must provide some point for doing so. Communication with the service is done using message passing. A useful first step in defining interaction with the service is to look at the phases specified in section III and determine suitable groupings of operations provided by an exchange service. This led to the specification of four areas of functionality between the customer and the service:

1) **Document Management** allowing customers to upload their documents and associated evidence of origin.
2) **Exchange Management** for starting and conducting exchanges of uploaded documents (either participant may start an exchange if they have permission).
3) **Service Notification** allowing customers to be notified of important events by the service (e.g. 'an exchange has been started' or 'an exchange has been successfully completed and evidence is now available').
4) **Evidence Retrieval** allowing customers to retrieve evidence (and exchanged documents) they are entitled to.

Note the deletion or modification of existing documents is not provided but could be supported as the a copy of the document at the time of exchange is stored as part of the evidence log.

These areas of functionality dictate that to interact with the service, the client requires:

1) **Outgoing Queue** for messages from the customer to the service.
2) **Incoming Queue** for messages from the service to the customer.
3) **Notification Topic** allowing the service to notify the customer of important events.

The use of two queues and one topic per customer fine grained access control at the boundary of the service.

The use of messaging in this manner requires a format for messages to be exchanged. A simple XML format was created for this work, similar to SOAP in that it comprises a header with arbitrary metadata and a payload for processing. The metadata will include information about which operation is being invoked (e.g. begin an exchange) and the relevant data to that request (e.g. the identity of the document to be exchange and the recipient). In the case of uploading a document, the payload could contain a complete RosettaNet or ebXML message, allowing compatibility with existing standards. All messages within the system have a unique message identifier, to handle potential for duplicate messages as a result of at least once delivery guarantee (in contrast to at most once and exactly once delivery offered by messaging such as JMS).

At this point, to interact with the service the customer must understand the messaging format used for communication. This may be acceptable for some customers, but the message based interaction can be easily masked through a client API. Four simple programmatic APIs were created to do this, one for each of the previously identified areas of functionality. These simple APIs are asynchronous and require only business documents or evidence from the customer, hiding the underlying message format.

B. Detailed Service Design

Section VI-A discussed interaction with the service, Figure 6 provides a closer view of the internals of the exchange service and details how the *In* and *Out* queues and *Event* topic connect to the internal service components. Solid interconnecting lines denote communication between the customer and the service. Dashed interconnecting lines denote message flow within the exchange service. Dotted interconnecting lines show notifications from components to the event topic. Details omitted in the figure for clarity will be subsequently explained.

An important clarification is that the *Work* queue within the service is also connected to the *Out* queue of B and the *Done* queue is also connected to the *In* queue of A. That is, all messages coming into the service are aggregated into the *Work* queue and all messages leaving the service are distributed from the *Done* queue. This allows the number of message processors within the service to be scaled arbitrarily to cope with the service workload.

A complete exchange between A and B would result in messages and notifications passing through both *In*, *Out* and *Event* destinations, and through the *Work* and *Done* queues within the service, as shown in the example in section VI-C.

All logic within the service is written using the Scala programming language [16], providing full compatibility with Java and its libraries. Communication within the exchange service (and thus the cloud-provider) is facilitated by the

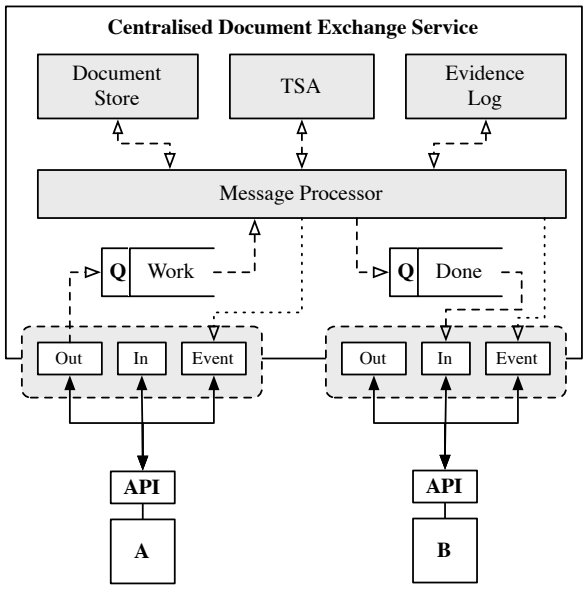


Figure 6. Exchange service internal components and message flow.

Scala actors package (supporting concurrency through message passing). Communication crossing the cloud-provider boundary is facilitated by messaging middleware.

The following Amazon Web Services provide cloud-based infrastructure used to construct the exchange service:

- **Simple Queueing Service** (SQS) for queue based messaging between customers and the service.
- **Simple Notification Service** (SNS) for event notification between customers and the service.
- **SimpleDB Service** (SDB) for a document database containing customer information and for flexible configuration storage for the service.
- **RelationalDB Service** (RDS) for hosting a MySQL database containing exchange information (and evidence).
- **Simple Storage Service** (S3) to store documents uploaded into the system by customers.
- **Elastic Compute Cloud** (EC2) to host and execute the logic of the service, combining components into the centralised exchange service.

Amazon has been chosen as the cloud provider for all component services simply for ease of use. They are assumed to be sufficiently trusted as a cloud provider. Documents, NRO and NRR evidences stored within the service in the cloud are communicated over encrypted transmission channels (using public/private key cryptography) meaning the private key is required to decrypt information obtained from the exchange service.

Each of the components used to construct the service can be scaled as needed to cope with increased loads. This could include provisioning more resources from SQS, SNS and SimpleDB to cope with more customers, increasing resources from the RelationalDB and Simple Storage services

to deal with more document uploads or spawning additional message processors by increasing the power of individual EC2 images, or deploying new ones.

C. Example Document Exchange

The following describes an document exchange between A and B.

Initially, customers A and B will both spawn their business processes, subscribing to their own *exchange notifications* from the exchange service. Customer A will then use *document management* to upload their document, D, and their NRO evidence to the exchange service. Upon completion of the upload, A invokes an exchange with B for D using *exchange management*. B receives notification through its *exchange notification* subscription that A has initiated an exchange of D and a digest of NRO to sign and return (i.e. the NRR) through *exchange management*. Upon receiving the signed evidence from B, the exchange service notifies both A and B the exchange is complete and allows them access to their respective documents and evidence through *evidence retrieval*.

The document upload, exchange invocation, notification to B, submission of NRR, release of D and NRO and notification of success to A and B are each achieved by a single message exchange between the service and client.

While the document upload and exchange were presented together in this example, they represent the sequential execution of all three phases as described in section III. Exchanges cannot be invoked unless a document and its evidence of origin have been uploaded. If B fails to provide NRR, the exchange would eventually time out and notify participants of such.

VII. Related Work

A primary concern when deploying services relating to trust is, even if the implementation of the document exchange service is written and verified as correct, what guarantees are there that the data or computations in the cloud remain secure? Santos et al. [17] propose a framework for securing execution of services within the cloud using a combination of Trusted Platform Modules and existing trusted computing techniques. Such a framework could be used to increase confidence in the trustworthiness and security of a document exchange service based entirely on cloud technologies.

Specifically relating to business to business agreements and contracts, Strano [18] has built on previously mentioned work regarding electronically modelling business contracts [3]. More recent work provides a vocabulary for modelling contracts in terms of sets of events, obligations, rights and prohibitions. These sets contain actions participants are expected to perform (obligations), allowed to perform (rights) and prohibited from performing and the events which allow the sets to transform as interactions progress (e.g. once a purchase order has been submitted in a single conversation, another may not be sent). The events used by this implementation are typically emitted by the business participants to the contract, making them possibly untrusted. The exchange service in this paper could be extended to emit specific events on behalf of customers, making the events trustworthy, assuming trust in the exchange service itself.

VIII. Conclusions

The aim of this paper was to design and implement a first step in cloud-based fair B2B document exchange and consider possibilities for constructing such a new service. To this end it has been successful and has also explored possibilities for partial leveraging of the cloud to facilitate migration or address specialised concerns.

The exchange service implemented allows consumers to upload documents, supports successful exchanges between correctly behaving participants, supporting timeouts during the *retrieval* phase of an exchange and allows any participant to retrieve all evidence to which they are entitled. The service does not yet support the aborting of an exchange in progress or finer grained acquisition of evidence (e.g. searching or ordering returned evidence).

The exchange service is constructed leveraging cloud hosted technologies for all aspects. All storage occurs using automatically scaled services (i.e. SimpleDB, RelationalDB and Simple Storage Services). The possibility to provision message based points of interaction per customer allows fine grained access control to be enforced. Amazon SQS and SNS state they allow an unlimited number of queues and topics to be defined per account, meaning scaling to any number of customers is theoretically supported. The subsequent aggregation of the incoming queues towards a single internal worker queue means the number of message processors dealing with exchanges may be scaled arbitrarily to cope with workload size. Compared specifically to the previous JMS and Web Services work, cloud technologies made aspects including provisioning of per-customer resources and spawning of multiple services much simpler, which should make the overall system more rapidly and easily scalable by comparison.

A shortcoming of the implementation arises due to the handling of an entire exchange by one message processor node and the lack of failure detection for these nodes. While the system allows an arbitrary number of message processor nodes to support multiple exchanges, it means the failure of any single node results in the failure of all exchanges associated with that node. The service could be developed to handle failure more gracefully as all evidence is persistently stored during the execution of an exchange, failure detection would need to be integrated into the handling of message processor nodes and nodes would have to be modified to automatically recover possible exchanges.

The non-repudiation protocol support is limited to providing timeouts during the *retrieval* phase of an exchange, ideally it also should support timeouts across all phases and the possibility of aborting an exchange before the point of no return. Cook et al [4] discuss the protocol constructs, evidence and constraints regarding aborting non-repudiation protocol execution.

A. Future Work

Immediate work with regards to the service would include the use of standards such as XMQP rather than Amazon SQS for messaging, support for aborting exchanges where possible and introduction of fault tolerance and exchange recovery into message processing nodes. Support for additional non-repudiation protocols within the service would address possible concerns with revealing information to the TTPs as discussed in [8].

Further work building on this richer internal support would involve decentralising the service, and would require the well-defined inter-service communication formats as discussed in section V.

Beyond the programmatic API provided, a RESTful interface where all documents, exchanges and evidence are represented as resources would be desirable. Similar to decentralisation this would be contingent upon well-understood media types. The principles of Hypertext as the Application Engine of State (HATEOAS) would be particularly applicable to the representation of ongoing exchanges as RESTful resources. This would be achieved by placing another layer at the boundary of the service, to deal with the incoming and outgoing HTTP requests.

Other work involves generalising the exchange service into a cloud-based, predicated message delivery service. The service could allow a set of predicates to be defined within the service (decoupled from the messages themselves). These predicates could specify conditions about specific senders, recipients, content types, message contents or events generated during an exchange. The delivery service would then be responsible for satisfying these predicates in the course of correlated business conversations.

Predicated delivery could be combined with Strano's work [18], subsuming the contract specification and checking implementation into the exchange service allowing interactions and exchanges to be driven by electronic contracts.

ACKNOWLEDGEMENT

This work is part-funded by the UK EPSRC and Red Hat UK Ltd under CASE/CNA/07/71: "Dependable Service Oriented Architectures for Business-to-Business Collaboration".

REFERENCES

[1] W. Baker, A. Hutton, C. Hylender, and C, "2009 Data Breach Investigations Report," Verizon Business RISK Team, Tech. Rep., 2009.

[2] J. Zhou and D. Gollmann, "Observations on Non-Repudiation," in *Advances in Cryptology—ASIACRYPT'96*, Kyongju, Korea, 1996.

[3] C. Molina-Jimenez, S. K. Shrivastava, and J. Warne, "A Method for Specifying Contract Mediated Interactions," in *Proc. 9th IEEE Int. EDOC Enterprise Computing Conf.*, Enschede, Netherlands, 2005.

[4] N. Cook, P. Robinson, and S. K. Shrivastava, "Design and Implementation of Web Services Middleware to Support Fair Non-repudiable Interactions," *Int. J. Cooperative Information Systems (IJCIS) Special Issue on Enterprise Distributed Computing*, vol. 15, no. 4, pp. 565–597, Dec. 2006.

[5] D. Mortimer and N. Cook, "A Declarative Approach to Configuring Business-to-Business Conversations," Newcastle University, Newcastle upon Tyne, Tech. Rep., 2009.

[6] OASIS, "OASIS ebXML Messaging Services Version 3.0: Part 1, Core Features," pp. 1–111, Oct. 2007.

[7] RosettaNet, "RosettaNet Overview: Clusters, Segments and Pips," pp. 1–122, Jan. 2009.

[8] S. Kremer, O. Markowitch, and J. Zhou, "An intensive survey of fair non-repudiation protocols," *Computer Communications*, vol. 25, no. 17, pp. 1606–1621, 2002.

[9] N. Asokan, "Fairness in Electronic Commerce," Ph.D. dissertation, Jun. 1998.

[10] H. Pagnia and F. Gärtner, "On the impossibility of fair exchange without a trusted third party," Dept. of Computer Science, TU Darmstadt, Tech. Rep. TUD-BS-1999-02, 1999.

[11] T. Coffey and P. Saidha, "Non-repudiation with mandatory proof of receipt," *SIGCOMM Computer Communication Review*, vol. 26, no. 1, Jan. 1996.

[12] J. Zhou, *Non-repudiation in Electronic Commerce*. Artech House Computer Security Series, 2001.

[13] R. Sion and M. Winslett, "Regulatory-compliant data management," *Very Large Databases*, pp. 1433–1434, 2007.

[14] AMQP, "AMQP A General-Purpose Middleware Standard," 2009.

[15] R. T. Fielding, "Architectural Styles and the Design of Network-based Software Architectures," Ph.D. dissertation, University of California, Irvine, 2000.

[16] M. Odersky, "Scala Language Specification 2.8," 2010. [Online]. Available: http://www.scala-lang.org/docu/files/ScalaReference.pdf

[17] N. Santos, K. P. Gummadi, and R. Rodrigues, "Towards Trusted Cloud Computing," in *HotCloud*. San Diego, California: Usenix, 2009.

[18] M. Strano, C. Molina-Jimenez, and S. Shrivastava, "A rule-based notation to specify executable electronic contracts," in *RuleML '08: Proceedings of the International Symposium on Rule Representation, Interchange and Reasoning on the Web*. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 81–88.