A Secure Dynamic Collaboration Environment in a Cloud Context*

Chris Piechotta[1], Adam Enø Jensen[2], Martin Grooss Olsen[3],
Joey W. Coleman[4] and Peter Gorm Larsen[4]
[1]*Mjølner Informatics A/S, Aarhus, Denmark*
[2]*Systematic A/S, Aarhus, Denmark*
[3]*Web HQ A/S, Aarhus, Denmark*
[4]*Department of Engineering, Aarhus University, Aarhus, Denmark*

Keywords: Cloud, Access Control, Security, Cloud Storage, Collaboration, Information Security.

Abstract: In recent years, the cloud has emerged as an attractive means for hosting and delivering services over the Internet. This has resulted in a renewed focus on information security in the case where data is stored in the virtual space of the cloud and is not physically accessible to the customer. This paper addresses the increasing security concerns of migrating to the cloud and utilising it for data storage, focusing on securing data in an untrusted cloud environment and ensuring detailed data access control in the cloud.

1 INTRODUCTION

In recent years the paradigm of *cloud computing* has been subject of much attention. The idea of dynamic resource provisioning to provide scalability and availability at relatively low cost makes cloud computing attractive for businesses. Cloud computing is usually partitioned into three primary services (Mather et al., 2009): Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS). Numerous other services exist although some with questionable justification. The term *Anything as a Service* (XaaS) covers all of them (Schaffer, 2009).

A common property of all the services is the lack of physical control of stored data, which hinders some potential customers from migrating to the cloud. This, combined with the Cloud Service Provider (CSP) having full access to stored data is a considerable concern and a primary hindrance for cloud migration (Zhou et al., 2010).

One possible use of a cloud solution is when different independent stakeholders wish to collaborate on a shared assignment (e.g. a consortium producing a joint bid for tender). Such a *dynamic collaboration environment* has security challenges in connection to rights about who can see what data, and with the dynamic nature of such a coalition where new partners may need to be included and others may need to be removed.

This paper presents two conceptual designs for a dynamic collaboration environment that, together, address these concerns. These designs are based on work from the first three authors' thesis (Piechotta et al., 2012). The aim is to provide a generic platform for sharing data without disclosing confidential information to unauthorised entities, while retaining the ability to control access to stored data at a user specified granular level. As any system can be described by its properties, the designs will be compared to related work and evaluated based on its properties. Both designs have been implemented in a publicly available prototype to establish their feasibility.[2]

After the introduction, Section 2 describes related work that has been used as a basis for the work reported here. Section 3 presents the model used to evaluate the various solutions, using properties that are particularly relevant to a collaboration platform. Sections 4 and 5 presents our core research results in the form of the two consecutive conceptual designs, and compares these against the related work. Finally Section 6 provides concluding remarks and indicates future work.

*All authors were with the Department of Engineering at Aarhus University when the research described in this paper was conducted.

[2]Available at https://github.com/iqman/MACMSC

Piechotta C., Jensen A., Olsen M., Coleman J. and Larsen P..
A Secure Dynamic Collaboration Environment in a Cloud Context.
DOI: 10.5220/0004357004060415
In *Proceedings of the 3rd International Conference on Cloud Computing and Services Science* (CLOSER-2013), pages 406-415
ISBN: 978-989-8565-52-5

2 BACKGROUND AND RELATED WORK

In this section we describe the related work on two cryptographic primitives that are fundamental to our conceptual designs. Afterwards two related designs that inspired our designs will be briefly presented.

2.1 Proxy Re-encryption

Asymmetric encryption (Simmons, 1979) provides the basis for Proxy Re-Encryption (PRE). PRE enables transformation of a ciphertext encrypted with one key into a ciphertext that is encrypted by a different key, all without ever decrypting it first and without revealing any information about the cryptographic keys (Blaze et al., 1998). This allows for PRE to be performed by a semi-trusted third party, called an *encryption proxy*.

Consider two persons Alice and Bob, each having their own cryptographic key pair[3]. Alice creates a special PRE token based on her private key and Bob's public key and gives this token to the encryption proxy. This allows the encryption proxy to transform a ciphertext encrypted with Alice's public key into a ciphertext encrypted with Bob's public key which, in turn, allows Bob to decrypt it using his private key.

Transitive and non-transitive PRE schemes exist (Ateniese et al., 2006). Consider three persons, Alice, Bob, and Carol that each have a cryptographic key pair, and the encryption proxy that has a token to transform a ciphertext from Alice to Bob and a token to transform from Bob to Carol. The transitive scheme then allows the proxy to transform from Alice to Carol by performing two transformations using both tokens.

However, PRE has the shortcoming that it is possible for a user and proxy to collude and reveal the private keys of other users (Dong et al., 2011).

2.2 Asymmetric Searchable Encryption

Asymmetric Searchable Encryption (ASE) is a cryptographic scheme offering confidentiality while allowing searching a ciphertext for keywords (Boneh et al., 2004). This makes the scheme usable in numerous scenarios and is considered more flexible with regards to expressiveness of search queries. However, the cryptographic operations used in ASE are relatively slow making it inefficient (Kamara and Lauter, 2010).

2.3 Efficient Searchable Encryption

Efficient Searchable Encryption (ESE) is based upon ASE, but improves upon its efficiency. ESE uses asymmetric encryption but deterministically generates a token for each keyword associated with the ciphertext (Bellare et al., 2007). Thus the tokens are suitable for use in a map of data entities and keywords. ESE supports Single Reader and Multiple Writers ($SR+MW$) scenarios.

2.4 Related Designs

Two distinct designs have been used as inspiration for our conceptual designs. Each has some strengths and weaknesses, so we seek to consolidate their strengths in our design while eliminating their weaknesses.

2.4.1 Two-step Encryption Scheme

The first inspiration is a design that uses the cloud to store encrypted data and leaves key and access management to a central entity called the *Data Owner (DO)* (Kamara et al., 2011).

The stored data is encrypted using a two-step encryption scheme. First, the data is encrypted with Symmetric Searchable Encryption (SSE), using a fresh randomly-generated encryption key. Then, that encryption key is encrypted using Attribute-Based Encryption (ABE) (Yu et al., 2010; Zhao et al., 2011). Finally, the encrypted data is stored in the cloud.

Users authorised to access data in the cloud are issued credentials matching their access rights by the DO. A separate token generator service is used to handle search requests and to issue the user a search token, which he sends to the cloud. Based on the search token, the cloud identifies the encrypted data that should be returned to the user.

The benefit of this approach is that the cloud only contains encrypted data, and the searchable encryption allows searching through encrypted data without downloading and decrypting it first, enchancing performance. Integrity is preserved by introducing a data verifier, usable only by the DO, to verify the integrity of stored data.

A severe drawback of this design is that data can only be put into the cloud when the system is initialised. The lack of support for Multiple Reader & Multiple Writer ($MR+MW$) scenarios by the employed encryption scheme is also problematic.

[3]A key pair consists of a private key kept by the user and a public key that can be publicly known to others, in this case the cloud.

Because the DO manages key distribution and cloud access he becomes a bottleneck, resulting in a negative effect on availability.

The tokens used to request data from the cloud are generated deterministically and could lead to breach of confidentiality since access patterns can be analysed through observation.

2.4.2 Trusted Key Management Server

This second inspiration is a design that, although it does not consider the cloud context, aims to achieve support for $MR + MW$ scenarios by combining ESE and PRE (Dong et al., 2008).

Keys are handled by a separate and trusted Key Management Server (KMS). The server's role is limited to issuing and revoking user access and thus access to the server is often not required.

The KMS is responsible for generating and storing a master key pair. When adding a user to the system, the KMS generates and issues a new key pair to the user and generates two PRE tokens used when uploading and downloading data, respectively.

When a user wishes to upload data, he associates the data with keywords, encrypts the data and keywords using his public key, and transmits the resulting ciphertext to the server. The encryption proxy transforms the data and keywords to be encrypted by the master public key by using the first token given by the KMS.

When downloading data, the user uses his public key to encrypt the keyword to search for and transmits the resulting search token to the server. The server transforms the search token so that it is encrypted by the master public key and then performs the search. The identified data is transformed so that it is encrypted by the user-specific public key using the second token generated by the KMS. Data is then transmitted to the user and decrypted using the user's private key.

As all users have distinct keys, a user can be efficiently revoked by removing the token to transform ciphertext between the master public key and the user's public key. Because there are two tokens used for reading and writing data respectively, the solution also supports coarse-grained access control as removing one of them will prevent a user from either reading or writing data.

3 EVALUATION MODEL

The ultimate goal of the dynamic collaboration environment is to ensure data security and fine-grained data access while retaining the benefits of utilising cloud storage. To facilitate an evaluation of the conceptual designs and provide sufficient comparison to the related designs, certain properties have been selected and will be the focus for the comparison. The chosen properties, presented in Table 1 and described in detail below, are the ones we consider the most important in terms of addressing the security issues preventing cloud migration.

Table 1: Overview of evaluation properties.

Category	Property
Security	Data confidentiality Data integrity Data availability Non-repudiation Auditability User-to-user anonymisation
Access control	User addition and revocation Fine-grained access control Access control delegation Many-to-many file sharing
Performance	Scalability Durability

Data confidentiality, Integrity, and Availability. The designs must enforce information-security. This is taken to be confidentiality, integrity and availability.

Non-repudiation and Auditability. To ensure the applicability of the designs in many domains, non-repudiation and auditability should be supported to mitigate errors and fraud.

User-to-User Anonymisation. To prevent potential cartelisation or collusion between entities involved in the collaboration environment, user-to-user anonymisation should be supported and controlled by the DO.

User Addition and Revocation. A collaboration environment is dynamic and involves adding new participants, while revoking others.

Fine-grained Access Control. Different levels of trust require different levels of access to information. Users often require access only to a strict subset of the resources in the system, and should not be allowed to access other parts.

Access Control Delegation. Having only one entity manage access control for all users creates a bottleneck. This necessitates the ability to allow multiple users to manage access at different levels.

Many-to-Many File Sharing. To ensure applicability in a wide range of usage scenarios the collabo-

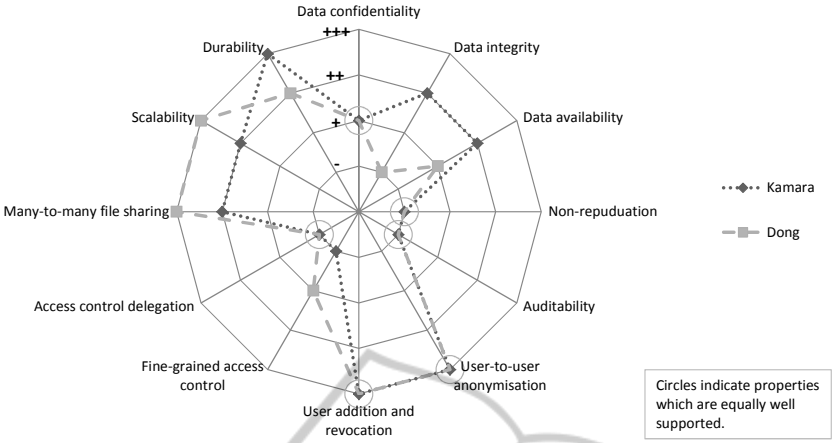

Figure 1: Comparison of designs in related work.

ration environment should support many-to-many file sharing.

Scalability. Due to the dynamic nature of a collaboration environment in terms of resource usage and number of users, it must support scalability.

Durability. When handling potentially large amounts of data, potential hazards such as hardware failures may result in data loss. This is unacceptable and thus the collaboration environment must possess high durability.

The different designs will be rated at four levels, starting with a minus, as the lowest score, and ending with up to three pluses, as the highest score. Each level on the scale has a clear definition (Piechotta et al., 2012). This makes distinctions between the different designs clear and makes an overall comparison of them possible. Figure 1 shows a comparison between the two designs in the related works and the defined properties.

4 SECURING DATA

This section describes *Secure Dynamic Cloud-based Data-sharing (SDCD)*, which comprises the core of our research, and the foundation of the dynamic collaboration environment. SDCD incorporates elements from the research and experiences of related work to form a dynamic collaboration environment as introduced in Section 1. Security assumptions are outlined in Section 4.1, an overview of the conceptual design and the terms important to SDCD is given in Section 4.2, also describing the operations of SDCD and providing a comparison to related work.

4.1 Security Assumptions

When cloud storage is utilised in a dynamic collaboration environment the benefits of the cloud are gained, however, this exposes the stored data to security risks. Data security may be compromised when data is stored in the cloud or during transmission to and from the cloud. Thus, when using cloud storage, these security risks must be mitigated.

When using cloud storage, the CSP is assumed to be *honest but curious*. This means that the CSP is expected to be interested in learning the contents of stored data, and has full access to everything stored in the cloud, but it will faithfully follow any protocol provided by the Data Owner. The CSP will not actively manipulate data or communications. The same is assumed for any user accessing the untrusted cloud. Furthermore, it is assumed that the CSP may attempt to collude with an arbitrary user to attempt to gain knowledge about the stored data or queries of any other user.

4.2 The Conceptual Design

Essential parts of SDCD are based on cryptography. In particular, PRE is combined with ESE to attain support for $MR+MW$ while being able to search for data based on keywords. To increase performance, SDCD applies hybrid encryption by combining symmetric and asymmetric encryption. This allows SDCD to encrypt the bulk of data using symmetric encryption with a randomly generated key and then encrypting the key using asymmetric encryption.

By combining PRE and ESE, SDCD ensures that all shared data is encrypted while maintaining support for individual keys for each user. Having individual keys makes it possible to efficiently revoke users by removing their corresponding PRE token from the

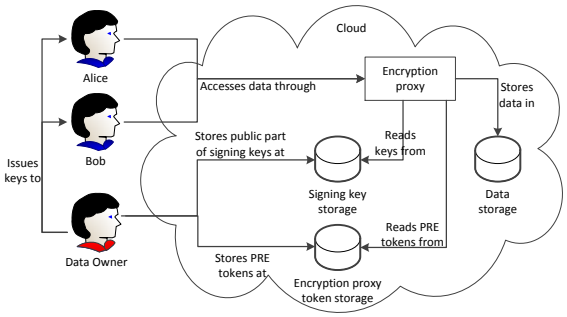

Figure 2: Conceptual overview of SDCD.

encryption proxy, thus rendering the respective user's key useless.

The concept of a DO is central to SDCD. The DO is responsible for generating keys and PRE tokens for all users. This puts the DO in complete control of users having access to shared data. Although the DO is central to the system, users access data directly from the cloud and users depend only on the DO for being granted the initial access. Figure 2 gives a conceptual overview of SDCD.

4.2.1 Data Entity

In SDCD, a single unit of data is called a *data entity*. A data entity encapsulates a resource that may be shared. It has an identifier, a non-empty set of attributes, and a payload. The identifier is randomly generated, but must be unique. The identifier is assigned to a data entity when the entity is created and remains unaltered throughout the lifetime of the entity. Each attribute corresponds to a single keyword and contains two fields. The first is the encrypted keyword. The second is an identifier for the attribute which is deterministically generated from the unencrypted keyword. The payload is the resource that should be shared. To support hybrid encryption, the data entity also contains a field for the key used by a symmetric encryption scheme. Finally, each data entity contains the signature of the author. Table 2 shows an overview of a data entity.

Table 2: Overview of the fields in a data entity.

Identifier		
Set of attributes	Attribute 1	Keyword 1
		Identifier of attribute 1
	⋮	....
		....
	Attribute *n*	Keyword *n*
		Identifier of attribute *n*
Payload		
Symmetric encryption key		
Signature		

4.2.2 Keys and Tokens

The DO owns a master key pair. All data entities stored in the cloud are encrypted using the master public key. Each user also has two key pairs: a PRE key pair and a signing key pair. The PRE key pair allows a user to decrypt data entities received from the cloud. The corresponding PRE token is placed in the cloud. The signing key pair is used by the user to sign data entities he creates or modifies. The signing public key is located in the cloud, and is used by the cloud to verify the integrity of data entities. All keys are generated and distributed by the DO. The entire key pair is distributed to the user and the public key to the cloud. In addition to the key pairs, each data entity has a unique symmetric encryption key. Finally, a search token is used when a user searches for data entities.

4.2.3 Data Entity Communication

When a user shares a resource, the data entity is created and encrypted in the user's environment. When the data entity subsequently is accessed, it is re-encrypted in the cloud and only decrypted in the receiving user's environment. This means that the data entity is never stored or processed in the cloud as cleartext. This scenario is depicted in Figure 3. In the figure, the steps are:

1. The user, Alice, prepares to upload a data entity by encrypting it with the master public key.
2. Alice uploads the data entity to the cloud.
3. The cloud uses Alice's PRE token to re-encrypt the data entity.
4. The re-encrypted data entity is sent from the cloud to Bob.
5. Bob decrypts the data entity using his own private key.

4.2.4 Verifying Integrity

To enable the cloud to verify the integrity of data entities, SDCD uses *digital signing*. A data entity is

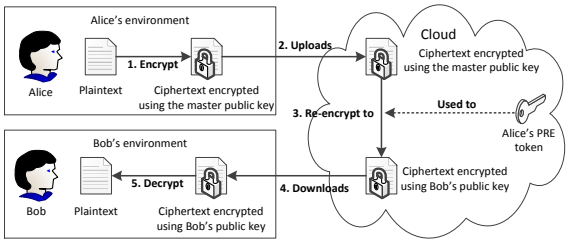

Figure 3: Overview of how data flows in SDCD.

signed by its author so the integrity can be verified in the future. The integrity verification scheme makes it possible for any authenticated user to request the cloud to verify the integrity of any data entity, independent of who the author was. This is accomplished while keeping all users anonymous to each other. To verify integrity, digital signing is used resulting in added support for *non-repudiation*.

The cloud must retain all public keys used for verifying signatures, even after a user has been revoked. This remains the case as long as a user is the author of any data entity. Furthermore, a user may have multiple signing keys if he has been added, revoked, and reinstated. However, from the user's perspective, only a single signing key exists as he is unaware of the multiple signing keys kept by the cloud.

4.2.5 Operations

The conceptual design offers support for a range of operations:

System Initialisation. The DO generates a master key pair and initialises the system by permanently taking the role of DO.

User Addition. To add a user, the DO generates a new PRE key pair, a new signing key pair and a single uni-directional PRE token for the user. The DO uploads the PRE token and the public signing key to the cloud. The DO gives the generated key pairs and the DO public key to the user.

User Revocation. To revoke a user, the DO removes the user's PRE token from the cloud. The user's signing key remains in the cloud for later verification of integrity of data entities where the user is the author.

Data Entity Creation. To share a resource, a user associates the resource with at least one keyword. A new data entity is generated with a randomly generated identifier and a randomly generated symmetric key. The resource is included in the data entity as the payload. Identifiers are generated for all attributes. The keywords inside the attributes and the payload are encrypted using the symmetric key which is then encrypted using the master public key. The user then signs the data entity using his private signing key.

The data entity is uploaded to the cloud which checks that the user is authorised to upload data by verifying the existence of the PRE token for the user. Finally, the cloud stores the data entity.

Data Entity Modification. To modify a data entity, a user performs the steps outlined in data entity creation. However, the identifier of the data entity to update is reused. The modifying user becomes the new author of the data entity.

Data Entity Deletion. To delete a data entity, a user sends the identifier of an existing data entity to the cloud and it is removed from the cloud storage.

Data Entity Searching. To search for data entities, a user enters a single keyword to search for. A search token is created from the keyword and sent to the cloud. The cloud finds data entities by matching the search token to identifiers of attributes for all stored data entities. The cloud re-encrypts matching data entities by using the user's PRE token. The matching data entities are then returned to the user who decrypts them using his private PRE key.

Data Entity Integrity Verification. To verify integrity, a user requests the cloud to perform the verification. The cloud verifies the integrity by using the public signing key of the user who is the author of the data entity.

Data Entity Author Authenticity Verification. The DO can use the cloud to determine and verify the authenticity of an author of a data entity. To prove the identity, the cloud performs an integrity verification using the public signing key of the user.

4.2.6 Comparing SDCD and Related Work

In contrast to some of the related work (Dong et al., 2008) SDCD is designed to work in a cloud context. The cloud context increases the potential for availability, scalability, and durability. When users are added to SDCD, no further interaction with the DO is required, effectively enabling users to share data entities even if the DO is not available.

Another important difference from the related work is that SDCD does *not* re-encrypt data when it is uploaded to the cloud. Instead the master public key is used directly by users. A benefit of this is reduced complexity and improved performance in the cloud when data is uploaded. Furthermore, re-encryption changes the binary layout of a data entity which would invalidate the signature added at the client. The disadvantage is that permissions to read and write cannot be distinguished. An authorised user in this design either has unrestriced access or none at all.

By using PRE, SDCD introduces the risk of collusion between the cloud and a user. However, the benefit of using PRE is support for many-to-many resource sharing which outweighs this concern. Another solution presented as related work (Kamara and Lauter,

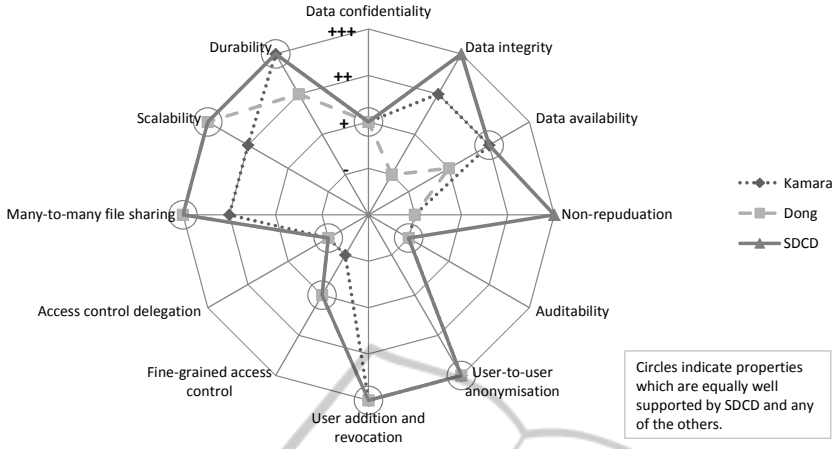

Figure 4: Comparison of SDCD to related work.

2010) does not share the concern for collusion, but it does not support many-to-many file sharing.

SDCD supports many of the important properties defined earlier. However, notably missing from SDCD is support for auditability and access control delegation. Figure 4 summarises and compares SDCD and the related work to the defined properties. SDCD meets or exceeds both related works on all points.

5 DETAILED DATA ACCESS

This section describes *Cloud-based Hierarchical Access Control (CHAC)*, comprising our research into extending upon SDCD, adding support for fine-grained data access control. Section 5.1 establishes the need for the access control model used in the design, and provides a detailed description of CHAC, explaining the basics of the access control model and its entities and operations. Section 5.2 presents details on user revocation in CHAC, while Section 5.3 provides a comparison between the original and the extended design.

5.1 Improving the Design

Fine-grained control of access to data is essential in any system where data is shared among users with differing levels of trust. To ensure data security, highly trusted users may be allowed full access while others are allowed only partial access to the shared data. Thus, effective management of these rights become important. To achieve this and to combine the two designs, we have designed CHAC as an extension of SDCD. Figure 5 shows how SDCD is extended through the addition of CHAC. Together they form a generic collaboration environment offering confidential cloud storage capable of enforcing fine-grained user access control and permission delegation through the use of a role hierarchy. This results in a transparent access control model with a clear set of rules defining how permissions are granted to users.

CHAC is based on the Role-Based Access Control (RBAC) model that originates from the NIST RBAC standard (Ferraiolo et al., 2001). The model is used to manage permissions that are used to access data stored in a cloud-based storage. The permissions are mapped onto roles that are assigned to users. The users can access data entities and perform selected operations through their assigned roles. The roles are related to one another in a hierarchical tree structure allowing inheritance of permissions while at the same time preventing child nodes from ever having permissions not held by their parent. Figure 6 shows the role relationships and how permissions are passed on down the tree structure.

The conceptual design must preserve the aspect of user-to-user anonymisation as this is an important security aspect with regards to collusion. CHAC has rules determining visibility of roles and users. A role can, given the right permissions, be used to see and

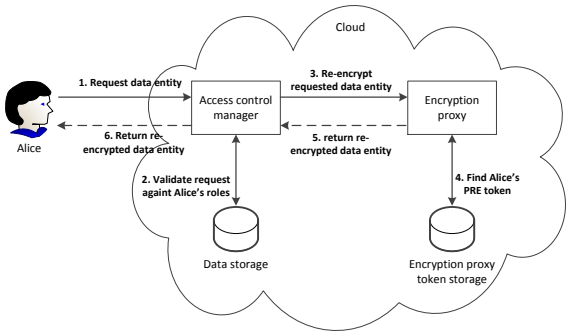

Figure 5: Conceptual overview of CHAC.

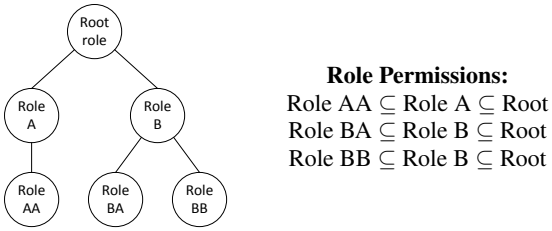

Figure 6: Overview of the role hierarchy.

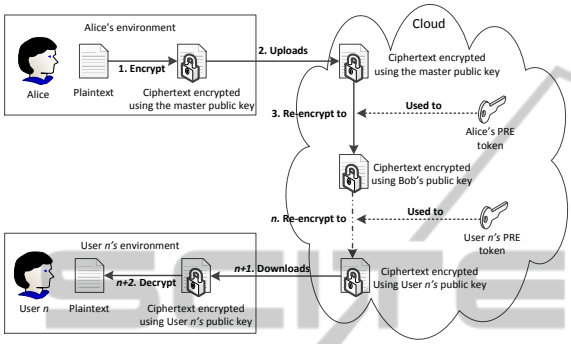

Figure 7: Re-encryption flow with n users.

manage all roles further down the tree on the same branch. That is, the roles sub-roles and their sub-roles, recursively. This also entails that a role can be managed by its parent roles, recursively. Thus, the root role can see and manage all roles and the users assigned to those roles in the tree. For any sub-role that can be seen, the users assigned to the role can also be seen. Users can have multiple roles assigned residing at different levels of the tree structure. For users to retrieve data from the cloud, encrypted data must be re-encrypted to match the respective user's cryptographic keys, since all data entities are encrypted using the DO's encryption key pair. This results in users having an indirect hierarchical relationship in terms of their cryptographic key pairs.

In the scenario where more users are subsequently added, data is re-encrypted multiple times as re-encryption of data is performed all the way down the indirect user hierarchy, formed through role assignment, to the user requesting data. Figure 7 shows how this affects the re-encryption process with n users. In the figure, Alice has added Bob who subsequently added another user ending with user n being added. When user n requests data, the data is first re-encrypted from being encrypted by the master public key to being encrypted by Bob's public key, using Alice's PRE token. Then it is re-encrypted to being encrypted by the next users public key, using the corresponding PRE token, until it is encrypted with user n's public key, enabling him to decrypt the data.

Users with sufficient permissions can manage roles, users, and data entities through various operations, such as creating new roles and users, or editing already existing roles and users. Data entities are stored separately from user and role information. This means that for each role that grants access to a set of data entities, a reference exists in the role for each data entity. Multiple roles can reference the same data entity. When creating a sub-role which has access to data entities of its parent, an additional reference is created for each of the data entities they share.

Auditability is introduced by allowing a root role to view actions performed by users. It can be used to review events associated with a specific role, data entity, or user. It can determine what has occurred, when it occurred, and who was responsible. Sufficient information must be stored in an audit log to ensure precise auditing. This means that whenever entities within CHAC are changed or new entities are created, these actions must be logged together with user information about who performed the action and through which role it was performed. The logging is an automatic procedure performed by CHAC and is invisible to users when they perform actions.

Audit logs are stored and maintained for users, roles, and data entities. Audit logs are kept even if an entity is deleted. This helps provide a complete picture of actions associated with even deleted entities and allows auditors to see when the entity was deleted and by whom. The audit logs are stored in the cloud storage in encrypted form using similar cryptographic techniques and keying schemes as the ones used to protect data entities.

5.2 User Revocation

The effort required to revoke users from the collaboration environment is minimal. This becomes evident when compared to other approaches towards user revocation that either require re-encryption of all shared data, or the maintanence of blacklists of revoked users to ensure that these users cannot gain access to data (Ruj et al., 2011). This issue is avoided through the role-based access control model used in CHAC. Revoked users are prevented from retrieving data from the cloud storage, and are unable to decrypt data, should they somehow get hold of it, as data needs to be encrypted under the revoked user's cryptographic key pair.

If data leaks should occur from somewhere in the role hierarchy, entire branches of the hierarchy can be suspended until appropriate actions have been taken.

In CHAC access control is performed in the cloud environment and does not require off-cloud services. This improves availability and durability compared to other approaches where the access control mech-

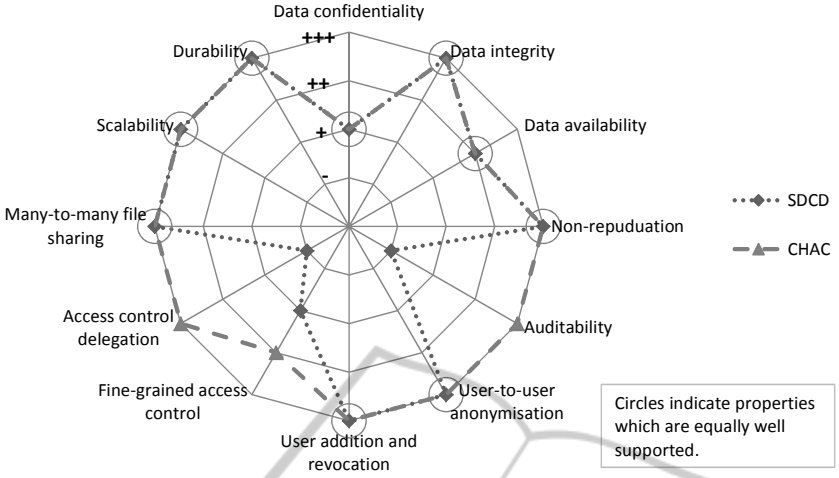

Figure 8: Comparison of CHAC to SDCD.

anisms are located off-cloud.

5.3 Comparing the Conceptual Designs

As shown in Figure 4, SDCD lacks the ability to delegate access control among users. For a dynamic collaboration environment to remain efficient, management of access control must be delegated among users. This ensures that no single user has the sole responsibility, thus becoming a bottleneck in the system. In contrast to SDCD, CHAC offers support for fine-grained access control for the dynamic collaboration environment. A comparison of the two conceptual designs can be seen in Figure 8. In addition CHAC introduces the ability of conducting auditability on entities used in the conceptual design. This is also an improvement to the initial design of SDCD.

6 CONCLUSIONS

Two new designs constituting a secure dynamic collaboration environment have been presented. The first design provides the foundation that ensures the confidentiality of data stored in the cloud while the second expands on the capabilities of the first to handle the dynamics of managing such a collaboration environment through detailed access control and access control delegation.

Our initial design employs a combination of symmetric and asymmetric cryptography. Data is encrypted with a master key generated by the Data Owner (DO) upon system initialisation. Integrity and non-repudiation is ensured by using digital signatures. All users have individual signing keys which are used when submitting data to the cloud. The initial design allows for integrity verification of stored data through digital signatures, and allows the DO to identify its respective author.

Proxy Re-Encryption (PRE) is applied to support many-to-many file sharing while retaining the ability to add and revoke users. This allows all users to submit data encrypted by the same key while receiving and decrypting data using a user specific key. Any user-specific key can be revoked without adversely affecting the other users, by removing a PRE token from the cloud. User-to-user anonymisation is achieved by restricting the capabilities of users to only encompass data access and data integrity checking. None of these operations reveal the identity of other users.

When a user has been granted access and has received his keys, no further communication between the DO and the user is required. This provides scalability and availability as the user communicates directly with the cloud.

Our improved design introduces a customised Role-Based Access Control (RBAC) model with SDCD at its core and enables support for fine-grained data access. An auditability scheme has been introduced, in addition to the access control model, adding the security aspect of auditing to the design.

CHAC is structured as a hierarchy, achieving a manageable structure that can easily be mapped onto job or usage functions in an organisational structure. Our initial designs had a user hierarchy, existing in parallel with the role hierarchy, to enable a structural dependency between users. However, this approach was deemed impractical due to the complexity of maintaining and continuously validating two different hierarchies with intertwining dependencies. To simplify the access control model, the user hierarchy was abstracted away and only the role hierarchy re-

mains, with the users now in a flat structure, or a *pool* of users. The access control model allows for delegation of access control management through clear role definitions.

The current life-cycle management of the PRE key pair hierarchy is based on the relationship between key pairs in the form of PRE tokens. To allow a user in a lower tier of the hierarchy to decrypt data, the encrypted data must first go through a chain of re-encryptions to pass down through the hierarchy ending at the user. As the collaboration environment evolves over time, the PRE key pair hierarchy potentially becomes very deep. This is undesirable as the additional re-encryptions result in an additional computational overhead and expose structural information about the hierarchy in the form of re-encryption duration. This information could be used for *timing attacks* which could potentially reveal cryptographic details on how data is protected (Kocher, 1996). This is a subject for future work.

REFERENCES

Ateniese, G., Fu, K., Green, M., and Hohenberger, S. (2006). Improved proxy re-encryption schemes with applications to secure distributed storage. *ACM Trans. Inf. Syst. Secur.*, 9:1–30.

Bellare, M., Boldyreva, A., and O'Neill, A. (2007). Deterministic and efficiently searchable encryption. In *Proceedings of the 27th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO'07, pages 535–552, Berlin, Heidelberg. Springer-Verlag.

Blaze, M., Bleumer, G., and Strauss, M. (1998). Divertible protocols and atomic proxy cryptography. In Nyberg, K., editor, *Advances in Cryptology - EUROCRYPT'98*, volume 1403 of *Lecture Notes in Computer Science*, pages 127–144. Springer Berlin / Heidelberg.

Boneh, D., Di Crescenzo, G., Ostrovsky, R., and Persiano, G. (2004). Public key encryption with keyword search. In Cachin, C. and Camenisch, J., editors, *Advances in Cryptology - EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 506–522. Springer Berlin / Heidelberg.

Dong, C., Russello, G., and Dulay, N. (2008). Shared and searchable encrypted data for untrusted servers. In Atluri, V., editor, *Data and Applications Security XXII*, volume 5094 of *Lecture Notes in Computer Science*, pages 127–143. Springer Berlin / Heidelberg.

Dong, C., Russello, G., and Dulay, N. (2011). Shared and searchable encrypted data for untrusted servers. *J. Comput. Secur.*, 19:367–397.

Ferraiolo, D. F., Sandhu, R., Gavrila, S., Kuhn, D. R., and Chandramouli, R. (2001). Proposed nist standard for role-based access control. *ACM Trans. Inf. Syst. Secur.*, 4(3):224–274.

Kamara, S. and Lauter, K. (2010). Cryptographic cloud storage. *Financial Cryptography and Data Security*, pages 136–149.

Kamara, S., Papamanthou, C., and Roeder, T. (2011). CS2: A semantic cryptographic cloud storage system. Technical report, Technical Report MSR-TR-2011-58, Microsoft Research, 2011. http://research.microsoft. com/apps/pubs. Accessed Apr 19 2012.

Kocher, P. (1996). Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In Koblitz, N., editor, *Advances in Cryptology – CRYPTO '96*, volume 1109 of *Lecture Notes in Computer Science*, pages 104–113. Springer Berlin / Heidelberg.

Mather, T., Kumaraswamy, S., and Latif, S. (2009). *Cloud security and privacy: an enterprise perspective on risks and compliance*. O'Reilly Media, Inc.

Piechotta, C., Jensen, A. E., and Olsen, M. G. (2012). Secure dynamic cloud-based collaboration with hierarchical access. Master's thesis, Aarhus University. Published as technical report ECE-TR-8.

Ruj, S., Nayak, A., and Stojmenovic, I. (2011). DACC: Distributed access control in clouds. In *Trust, Security and Privacy in Computing and Communications (TrustCom), 2011 IEEE 10th International Conference on*, pages 91 –98.

Schaffer, H. (2009). X as a service, cloud computing, and the need for good judgment. *IT Professional*, 11(5):4–5.

Simmons, G. J. (1979). Symmetric and asymmetric encryption. *ACM Comput. Surv.*, 11(4):305–330.

Yu, S., Wang, C., Ren, K., and Lou, W. (2010). Achieving secure, scalable, and fine-grained data access control in cloud computing. In *INFOCOM, 2010 Proceedings IEEE*, pages 1–9. IEEE.

Zhao, F., Nishide, T., and Sakurai, K. (2011). Realizing fine-grained and flexible access control to outsourced data with attribute-based cryptosystems. In Bao, F. and Weng, J., editors, *Information Security Practice and Experience*, volume 6672 of *Lecture Notes in Computer Science*, pages 83–97. Springer Berlin / Heidelberg.

Zhou, M., Zhang, R., Xie, W., Qian, W., and Zhou, A. (2010). Security and privacy in cloud computing: A survey. In *Semantics Knowledge and Grid (SKG), 2010 Sixth International Conference on*, pages 105–112.