A Cryptographic Protocol for Communication in a Redundant Array of Independent Net-storages

Martin Gilje Jaatun* Gansen Zhao†, and Stian Alapnes‡
*SINTEF ICT, Norway
Email: martin.g.jaatun@sintef.no
†South China Normal University, China
Email: gzhao@scnu.edu.cn
‡Telenor Corporate Development, Norway
Email: stian.alapnes@telenor.com

***Abstract*—This paper describes a cryptographic protocol for storing and processing data in a Cloud Computing setting, where users need not place absolute trust in the various Cloud Processing providers. This is achieved by distributing data among various Cloud Storage providers in such a manner that an individual data item does not divulge useful information about its owner, and only re-assembling data when it needs to be processed or returned to the user.**

I. Introduction

When computation and storage are outsourced to the Cloud, a cloud computing provider currently gets access to all data belonging to their customers. Due to the longer trust chain [1], and potentially large physical distances between customers and providers, customers' security concerns are in many cases scaring them away from the Cloud, thus preventing them from reaping the benefits of Cloud Computing. It is easy to imagine many consumers being uneasy about what huge corporate Cloud providers might want to do with their information, and even if the providers basically are honest, customers may still be exposed to threats such as insiders at the cloud provider.

The Redundant Array of Independent Net-storages (RAIN) was introduced by Jaatun et al. [2] as an approach to increased confidentiality control in cloud computing, by splitting up information in sufficiently small chunks, and distributing these among several storage providers, as illustrated in Figure 1. Furthermore, when data needs to be processed in any way, this is done in the cloud by re-assembling the minimal amount of data that is necessary, and then re-distributing the results. Thus, if the data chunks are small enough, and if it is hard for anyone but the owner to re-assemble them, confidentiality can be achieved without encryption, and without having to trust each individual cloud provider. The original paper [2] left a lot of open questions, and in the following we will tackle one specific challenge: How can we devise a protocol that allows the user to put data in the cloud, without allowing the individual providers to trace the information back to the user?

II. Related work on anonymous storage

The Free Haven project [3] describes a collaborative distributed storage system, where participants are allowed to store (or publish) data by offering to store data for others, in the same general fashion of peer-to-peer file sharing. The Free Haven project does not provide a new solution for the anonymous communications channel, but uses a set of anonymous remailers as a basis. The Free Haven project makes no assumptions on the participants being honest, but uses a reputation system to identify non-cooperative (or dishonest) nodes.

The OceanStore [4] system is also based on distributed storage, but is less concerned with ensuring anonymity of the individual users.

The ShareMind framework [5] [6] offers distributed privacy-preserving[1] computations, based on the principles of secure multiparty computations. Sharemind seems less focused on (anonymous) storage than computations; the current prototype solution is based on distributing data from one source among three nodes referred to as *data miners*, and is only secure as long as the three miners do not collude.

III. Preliminaries

This section will set out limitations for our contribution, and will detail security assumptions, principals involved and claims made on behalf of the solution.

A. Limitations

This paper will specifically not discuss many of the open issues in [2], such as how to deal with small information chunks that despite their size contain too much sensitive information, and will also not delve into problems related to the process of splitting[2] the data. For now, we can consider that the dataset is a random string of bits, which implies that each chunk of data is equally random, and has no relation with any other chunk. We can then concentrate on the challenge of preventing an adversary from tracking each chunk of data.

B. Assumptions

The RAIN solution makes the following security assumptions:

[1]It may be a matter for debate whether the solution rather should have been referred to as *confidentiality*-preserving.

[2]In other work, we will pursue how Rabin's work on splitting data [7] can be employed for the larger solution, but that is outside the scope of this paper.

1) We have a file (or dataset) that has been divided into small chunks
2) A provider will not be able to link two different chunks of the same dataset, should it gain access to them
3) The cloud service providers can be classified as "Honest but curious", i.e., we expect them to carry out the protocol faithfully, but they may try to access the information either through collusion or other means.
4) We have at our disposal an anonymous sender/receiver framework for communication in the cloud
5) There are enough simultaneous users to make anonymity feasible[3]
6) We have a lightweight authentication mechanism which can be used to regulate access to a data item
7) The C&C node has a list of "honest" Cloud Processing providers, and their public keys
8) The C&C node maintains a record of all data IDs
9) The C&C node maintains a record of all nonces generated by legitimate users and itself, for as long as a data ID is active
10) The user maintains a log of any outstanding requests sent to the C&C node, and will reject any unsolicited responses
11) The C&C node maintains a log of any outstanding requests sent to cloud processing providers, and will reject any unsolicited responses

The adversarial model is less powerful than Dolev-Yao, in that we assume that an adversary can observe all traffic, and possibly insert traffic, but not in general *delete* traffic (e.g. does not carry the message). As already stated, we also assume that the cloud providers are honest but curious.

C. Principals

Our world consists of the following principals:

- The User
- The C&C cloud service (named after the Command & Control node in a botnet)
- Multiple cloud storage providers
- One or more[4] cloud processing providers

Behind the scenes there are also some additional principals:

- An "IRC" cloud multicast service
- Multiple cloud storage agents
- Multiple cloud processing agents
- Multiple cloud mix-net agents

These are not discussed further in this paper.

D. Claims

- No cloud storage provider can associate any chunk of data with its owner
- No cloud storage provider can associate any chunk of data with another chunk of data from the same dataset
- No cloud processing provider can associate any dataset with its owner
- No adversary can delete or modify any part of a user's dataset

IV. Solution

We present a simplified solution, as illustrated in Figure 1. We have n Cloud Processing Providers $(CP_1, \ldots, CP_n)$ and m Cloud Storage Providers $(CS_1, \ldots, CS_m)$. As mentioned, we assume the existence of an anonymous sender/receiver framework, and when sending anonymous messages we will use the notation $\{destination, \ldots\}_{anon}$. Note that since the sender is supposed to be anonymous, there is no point in including the originator address in these messages. In the following we will examine how data is stored, retrieved and processed using a request-response protocol in the RAIN.

A. Protocol for Storing Data

We have [9] that D is a piece of data to be split and stored in a cloud, and an unspecified function splits D into a sequence H of smaller segments such that $H = (h_1, h_2, ..., h_n)$.

The user U will send the data D and its ID to the C&C node, encrypted with that node's public key:

$$U \rightarrow C\&C : \{store - full, auth, ID_D, D\}_{K_{C\&C}}$$

Here, "auth" is an authentication token used to verify the user's rights to the dataset[5]. If this message is replayed, it will be ignored; the only way to re-use a data ID is to first delete the data-set that uses it.

The C&C node performs the split [9] such that H can be represented as $H =< D_s, R_s >$ where

- $D_s = \{d_i | i = 1, ..., n\}$
- $R_s = \{< d_i, d_{i+1} > | i = 1, ..., n-1\}$.

Here, R_s specifies how the segments are related to each other; this knowledge is necessary for reassembly. We need to assign a unique ID (or pseudonym) to each data item, which we in the following refer to as ID_{di}.

The C&C node then distributes H among the cloud storage providers by assigning their respective identifiers $(CS_x, \ldots, CS_y)$ to the corresponding d_i, and sending the chunks anonymously:

$$\forall i | C\&C \rightarrow CS_j : \{CS_j, store - part, auth, ID_{di}, d_i\}_{anon}$$

The C&C node needs to maintain a table mapping which data items have been sent to which cloud storage service.

[3]This is a rather fuzzy assumption, but it is clear that if there are only two parties communicating, an adversary can trivially determine that all data observed leaving one party will arrive at the second party. This assumption of "more than a few" users is also used in, e.g., TOR [8]

[4]In our examples we only show one cloud processing provider, but we assume that there would be several provides to choose from, in order to avoid putting all our processing eggs in one basket.

[5]In the current description, we make no attempt to hide the identity of the user from the C&C node, so here we could assume that the authentication is an (as yet unspecified) conventional mechanism authenticating the user to the C&C provider. However, in the following we will use this authentication mechanism to control access to individual data items, and in this case the cloud providers should of course not know the identity of the user or other actors

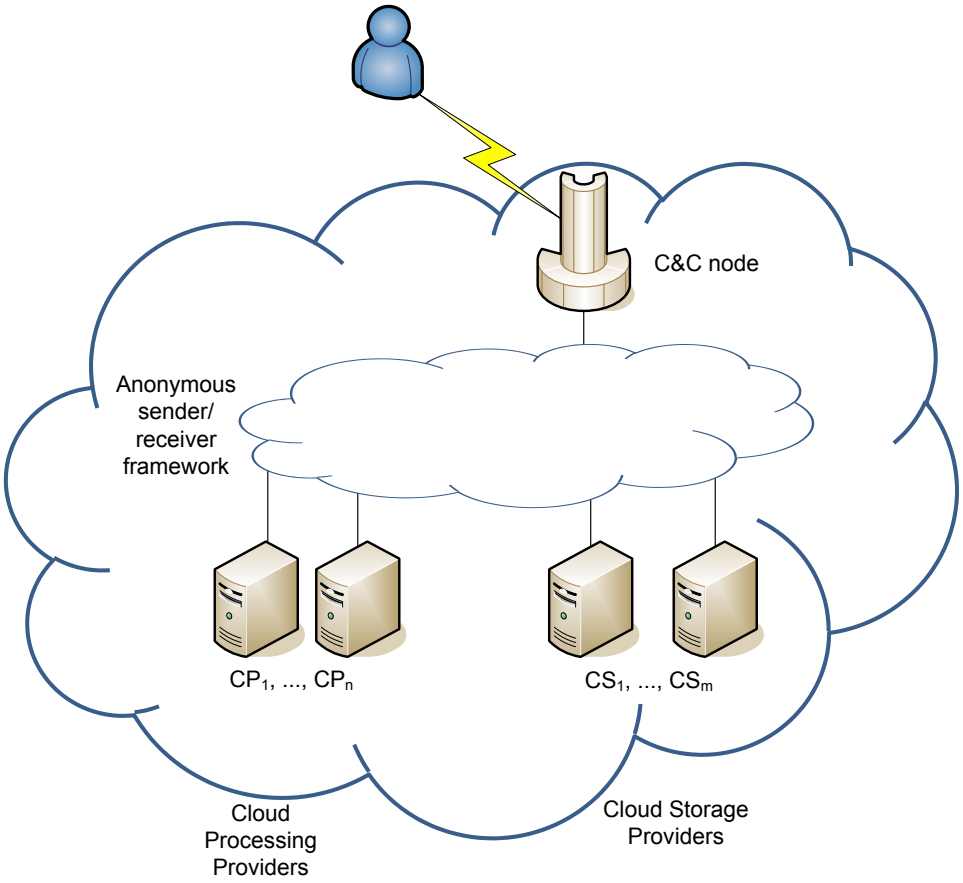


Fig. 1: Simplified RAIN scenario

Furthermore, it is important that the pseudonyms are unique within a given C&C provider, but created in such a manner that it cannot be determined that two different pseudonyms refer to data items from the same file. As mentioned in Section III-B, we're assuming that the traffic volume will contribute to hide which items belong to which datasets, so although we are effectively broadcasting the mapping tables, this should not matter: An adversary can tell that the data item with pseudonym *X* is stored with cloud storage service *Y*, but this information is of little use if there is no way to tie the pseudonym to a dataset (or user). Furthermore, since we are assuming an anonymous sender and receiver framework, the identity of the cloud storage provider is effectively a pseudonym as well.

B. Retrieving Data

The user may ask the C&C to retrieve a dataset:

$$U \rightarrow C\&C : \{retrieve - full, auth, ID_D\}_{K_{C\&C}}$$

When asked to retrieve a dataset, the C&C node will need to ask each storage service to return their respective data items:

$$\forall i | C\&C \rightarrow CS_j : \{CS_j, retrieve - part, ID_{di}\}_{anon}$$

Note that we do not need to authenticate when retrieving individual data items in order to fulfil any security claims made by RAIN.

Each Cloud storage provider[6] responds with its piece of the puzzle:

$$\forall i | CS_j \rightarrow C\&C \ : \{CS_j, return - part, ID_{di}, d_i\}_{anon}$$

The C&C node then re-assembles the data, and either returns it to the user:

$$C\&C \ \rightarrow U : \{U, return - full, ID_D, D\}_{K_U}$$

or sends it off to be processed as explained in the next section.

[6]It may be debatable whether it makes sense to include the provider ID in the response, but certainly the C&C node ID cannot be there, as it is supposed to be anonymous.

C. Processing Data in the Cloud

When the user wants to do something with the data, it will tell the C&C node:

$$U \rightarrow C\&C : \{process{-}cnc, operation, auth, ID_D, N_u\}_{K_{C\&C}}$$

Here, "operation" identifies what should be done, ID_D identifies the dataset, and N_u is a nonce chosen by the user.

The data will first have to be retrieved and re-assembled as explained above. The C&C node then selects an appropriate number of cloud processing providers, depending on the type of data and what is to be done with it. If the data is, e.g., a digital image, and the user wants to manipulate it using a Cloud-based image editor, then the complete data set typically needs to be sent to a single processing provider.

$$C\&C \rightarrow CP_j :$$

$$\{\{CP_j, process - cp, operation, D, K_{PC}, N_c\}_{K_{CPj}}\}_{anon}$$

The data, the nonce chosen by the C&C node, a symmetric key K_{PC} for encrypting the response, and the rest is encrypted with the public key of the cloud processing provider, and sent through the anonymous sender-receiver network.

$$CP_j \rightarrow C\&C \ : \{\{result - cnc, D_{result}, N_c\}_{K_{PC}}\}_{anon}$$

Note that since the C&C node keeps track of requests to processing providers, the operations are idempotent; replayed responses are ignored, and in case of response failures, a new request will be sent, canceling the former.

The result is returned to the user:

$$C\&C \ \rightarrow U : \{U, result - user, D_{result}, N_u\}_{K_U}$$

Again, the user will reject any spurious responses with a nonce that doesn't match that of an outstanding request.

If the result is a change in the dataset, it will either have to be re-stored or delivered to the user, depending on the user's wishes. If data items need to be updated or deleted, the authentication mechanism comes into play again. In any case, a confirmation is sent to the user, closing the outstanding request.

An example of an editing operation is shown in Figure 2. In this case, an image of a rodent (Figure 2a) is to be modified to become a cat (Figure 2d). This example also highlights an optimization opportunity; Figure 2b and 2c identify the modified areas of the image, and on completion only these parts need to be re-stored. The exact mechanisms of how to determine which parts have been changed are beyond the scope of this paper, however.

V. Implementation Considerations

Space does not permit a full implementation specification, but in the following we will illustrate in a little more detail how the actual storage and retrieval process may be realized from the C&C node's point of view.

Although we do not go into specifics here, it is clear that the actual splitting must depend on the type of document. The process is illustrated in Fig. 3. A user (or a client running e.g. in a cloud environment) initiates writing of content to the system. The user can configure which storage providers to use for certain file types or content. Part of the config contains information on how each of the storage providers can be used, i.e. description on how to access, write and read content. Typically this can be a proprietary web API. Based on the selection of storage providers available and the content type a recipe is generated. The recipe states the size of blocks the original file is to be split into and a sequence for writing the blocks to the various storage providers. Based on the recipe the content is divided in blocks that each is stored at a storage provider. The recipe is stored, and using the recipe, the content can be retrieved from the storage providers and assembled. The fileID is returned to the initiating part.

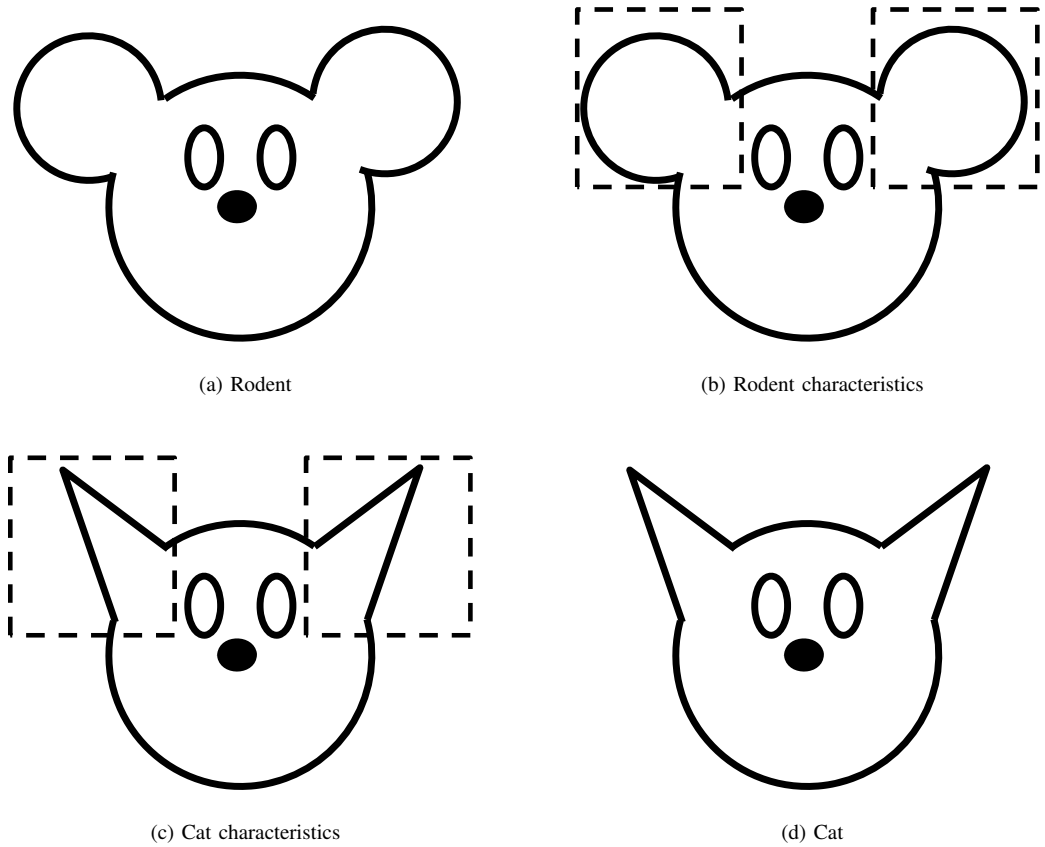
The retrieval process is illustrated in Fig. 4. A user (or a client running e.g. in a cloud environment) initiates reading of content from the system. The recipe is retrieved based on the fileID. Based on the recipe the file is read from storage providers and assembled. The assembled file is returned to the initiating party.

VI. Security Analysis

As can be seen from Table I, pain has been taken to avoid reusing commands that otherwise might have made it possible to replay messages from A to B as a message from B to C. This is according to Principle 1 of Abadi and Needham [10], and to some extent also Principle 3, since it ensures that every message will only be handled by the same type of actor as intended. However, full adherence with Principle 3 may be difficult to achieve in a setting of anonymous communication. The explicit naming of commands is also in accordance with Principle 10, since it allows for unambiguous encoding of each message.

Since data is sent encrypted from the C&C node to the Cloud Processing provider, it cannot be observed by an adversary, who cannot determine the symmetric key used to encrypt the response, and thus cannot do a suppress-replay attack to replace the result with a bogus result. We are assuming the C&C node has verified public keys to the providers, which means only the selected provider sees the data, but as long as the relationship between user and data is kept secret, it does not really matter exactly *which* Cloud Processing provider handles the data.

By applying the Scyther tool, we find (unsurprisingly) that the assumption that data and session keys from the C&C node are kept confidential holds, but unless the public key of the processing provider has been verified, we cannot assume that it remains confidential. Since the Scyther tool does not support



(a) Rodent

(b) Rodent characteristics

(c) Cat characteristics

(d) Cat

Fig. 2: Illustration of editing an image of a rodent to get a cat

Command	Explanation
store-full	Command from User to C&C to store a complete dataset
store-part	Command from C&C to Cloud Storage provider to store a piece of data
retrieve-full	Command from User to C&C to retrieve a complete dataset
retrieve-part	Command from C&C to Cloud Storage provider to retrieve a piece of data
return-part	Cloud Storage provider is returning a piece of data
return-full	C&C node is returning a complete dataset to User
process-cnc	Command from User to C&C to perform processing operation on a dataset
process-cp	Command from C&C to Cloud Processing provider to perform processing operation on a dataset
result-cnc	Cloud Processing provider returning result of processing operation on a dataset to C&C node
result-user	C&C node returning result of processing operation on a dataset to User

TABLE I: Summary of all protocol commands

verifying privacy/anonymity claims, it cannot be used to verify the full protocol.

Since we assume the presence of the anonymous sender/receiver framework, we will not attempt to prove any properties related to this. Also note that we have made no claims with respect to resource consumption on the cloud providers. Thus, it may be possible for a malfeasor to waste the resources of a Cloud Processing provider by replaying `process-cp` messages from the C&C node. This could be countered by having the Cloud Processing provider store all nonces N_c, and discard all messages with non-fresh nonces, but since this does not contribute to keeping data and users anonymous, it has been omitted to avoid forcing the providers to maintain state information. However, in a possible future commercial solution, this might be solved as part of a payment solution.

From a complexity point of view, assuming that there are in total n pieces of data stored in the cloud, let a malicious user try to illegally access a file, which has been split into k pieces and kept in the cloud. The malicious user must first re-assemble the whole file, taking two steps.

1) Step 1: All k pieces must be retrieved corrected out

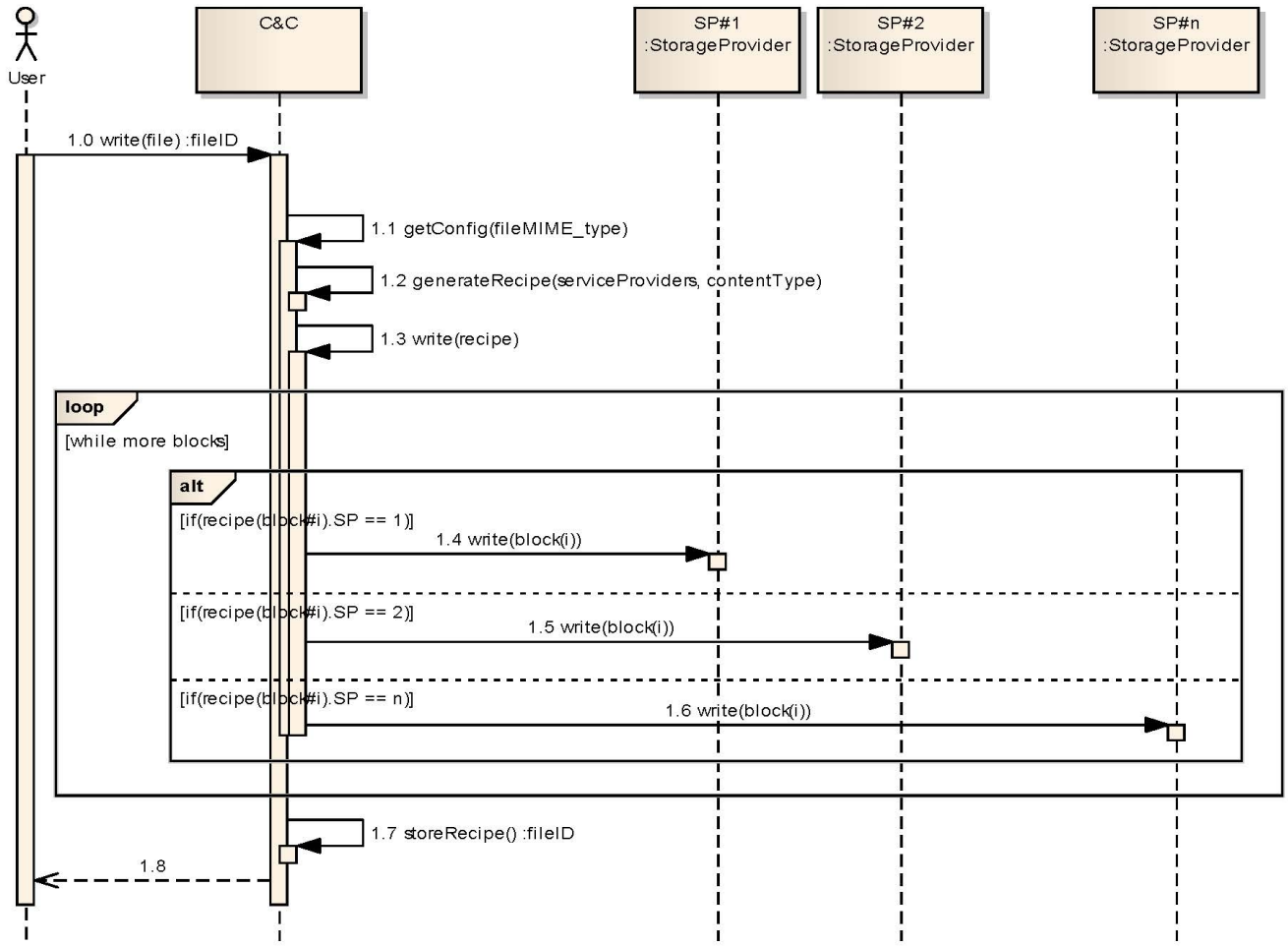


Fig. 3: Sequences for splitting content/file into blocks and writing to a set of storage providers

from the n pieces. The probability to retrieve the correct pieces is as follows.

$$p_1 = \frac{1}{C_n^k} = \frac{k!}{n*(n-1)*\ldots*(n-k+1)}$$

2) Step 2: Re-order all the k pieces into the correct order, given the k pieces. The probability of putting all k pieces in the right order without any knowledge of the original data is

$$p_2 = \frac{1}{P_k^k} = \frac{1}{k!}$$

Hence, the probability of re-assembling the file correctly is

$$p = p_1 \times p_2 = \frac{k!}{n*(n-1)*\ldots*(n-k+1)} \times \frac{1}{k!}$$

$$= \frac{1}{n*(n-1)*\ldots*(n-k+1)}$$

Assuming that there is a very large number of pieces in the cloud and each file is split into small enough chunks, n and k are both large enough to ensure that the probability p is small enough to counter attacks.

The cost for an attacker is far from only the computation complexity of re-assembling the k pieces. Due to the distributed characteristics of the proposed storage system, the system contains a very large amount of data and the data are distributed across various networks. The attacker attempting to re-assemble a file by brute force will have to have an extremely large storage space to keep all the retrieved data pieces (both the correct ones and the wrong ones), and it also has to afford the cost for the network bandwidth to transfer such an amount of data across the network.

VII. Discussion

For completeness, we mention that a full solution also needs to ensure the following assumptions:

1) A provider will not be able to gain any useful information from a single chunk of data
2) Outside means must ensure that a single provider does not receive enough chunks to re-assemble (parts of) the original data.

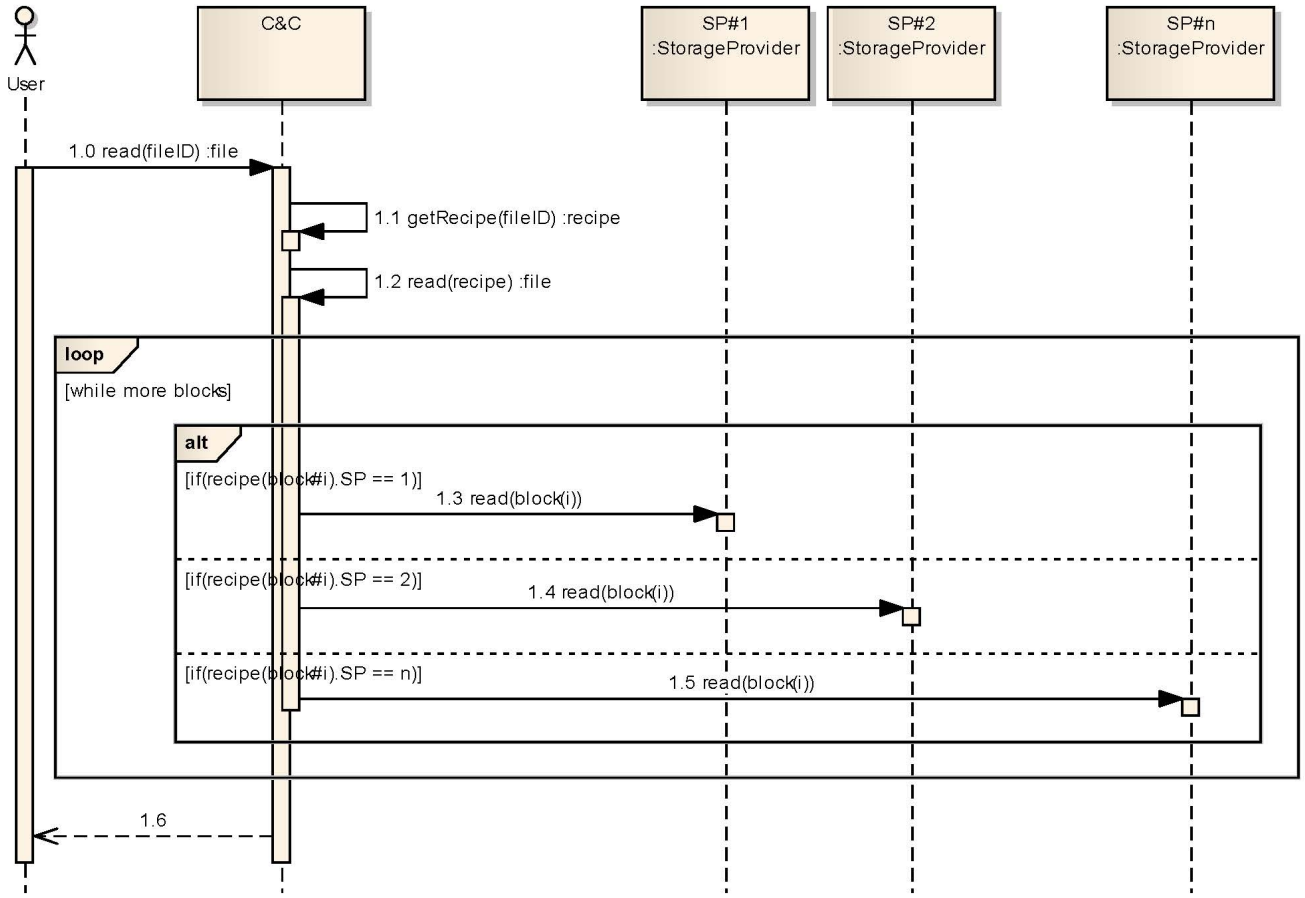


Fig. 4: : Sequences for retrieving blocks of content from a set of storage providers and assembling them into the original content.

As mentioned in Section III-B, the protocol would presumably be easy to break if there is only one user and one C&C node - but in that case there also would not be a need for the system. The popularity of such varied solutions as TOR [8], Polippix [11], and anonymizer.net seem to indicate that there will always be a certain minimum of users sufficiently concerned with privacy and anonymity to actively choose solutions like RAIN, if they are made available. We have not formally evaluated thresholds for number of users and/or amount of data to prevent an adversary (in practical terms) from reconstructing data; this remains an area for further study.

In periods of low traffic, a possible additional measure may be to distribute chaff, as advocated by Rivest [12]. Since this is one of the things we expect the anonymous sender/receiver framework to handle, we will not discuss it further here.

A. Searching and Indexing

One major unsolved problem with our solution is related to searching and indexing. Even if it were possible to create an index to search in, where can we store it? Thus, we currently have to accept that searching is not possible without re-constructing each file first.

B. Business model

It's been said that everybody wants security, but nobody is willing to pay for it. This means that not only is it difficult to get funding for security measures in organizations where security is viewed as a net expense, but most users are also not willing to put up with the extra inconvenience that added security mechanisms often imply. Contrasting this with the (at least currently) free services such as Google Docs [13] that cloud providers are throwing at customers, it may be hard to imagine anybody paying money to get the same thing "more secure".

However, privacy is evidently an issue for some people, as the usage statistics of the TOR network can testify [8], and also experiments in Scandinavia have shown that many people will choose privacy if it's made available to them [11]. In general, it is dangerous to confuse the concepts of "privacy" and "confidentiality", but in our case we believe that the privacy aspects will be the major driver for people wanting to keep their data confidential.

How to pay for the services anonymously has not been completely resolved. Most existing solutions such as TOR

[8] and Free Haven [3] are based on volunteer or barter arrangements, where participants get free use of the service by supporting parts of it on their own systems. The payment problem is also the main obstacle for Chaum's approach [14], since he assumes the existence of a digital cash system, something which remains elusive. Still, since the cloud paradigm is oriented toward *pay per use*, we believe it will be easier to solve this in the clouds than in many other situations. One option for further study would be for users to somehow pay for other users' cloud use; this is analogous to the volunteer or barter concept.

C. Trust

The C&C node will remain as a "single point of trust" as long as it is realized as part of the cloud. However, we maintain that even if we still have to trust the C&C node in the cloud, this in an improvement over handing all our data over to Google. The C&C node in effect plays the role of a Trusted Third Party, and generally does not need to have the enormous resources of the current commercial cloud providers. Thus, the C&C node could in principle be run by some small company in the user's neighbourhood, enabling a traditional trust relationship.

VIII. Conclusions

This paper has presented a cryptographic protocol for improved confidentiality when storing and processing data in the cloud. The solution is based on the rather large assumption of having an anonymous sender/receiver framework in place, the specification of which is left for further work.

An obvious area of further work is to create a proof-of-concept prototype, and this will be our next step. Other opportunities for further work include the design of a payment solution that would enable users to pay anonymously for RAIN service, and solutions for avoiding having to trust a C&C node in the Cloud, which could finally deliver on the promise of deliverance from trust. We have also not attempted to specify the authentication mechanism that would prevent unauthorized modification or deletion of information; further study is required to determine if an existing mechanism would suffice, or if a new mechanism is called for.

Acknowledgments

This work has been supported by the Telenor-SINTEF research agreement.

References

[1] K. Bernsmed, M. G. Jaatun, P. H. Meland, and A. Undheim, "Security SLAs for Federated Cloud Services," in *Proceedings of the Sixth International Conference on Availability, Reliability and Security (AReS 2011)*, 2011.

[2] M. G. Jaatun, Å. A. Nyre, S. Alapnes, and G. Zhao, "A Farewell to Trust: An Approach to Confidentiality Control in the Cloud," in *Proceedings of the 2nd International Conference on Wireless Communications, Vehicular Technology, Information Theory and Aerospace & Electronic Systems Technology (Wireless Vitae Chennai 2011)*, 2011.

[3] R. Dingledine, M. J. Freedman, and D. Molnar, "The free haven project: Distributed anonymous storage service," in *Proceedings of the Workshop on Design Issues in Anonymity and Unobservability*, 2000.

[4] S. Rhea, P. Eaton, D. Geels, H. Weatherspoon, B. Zhao, and J. Kubiatowicz, "Pond: the OceanStore Prototype," in *Proceedings of the 2nd USENIX Conference on File and Storage Technologies (FAST '03)*, 2003.

[5] D. Bogdanov, S. Laur, and J. Willemson, "Sharemind: a framework for fast privacy-preserving computations," Cryptology ePrint Archive, Report 2008/289, 2008, http://eprint.iacr.org/.

[6] "Cybernetica news blog - sharemind," 2008, http://research.cyber.ee/sharemind/, visited: Sept. 9, 2010. [Online]. Available: http://research.cyber.ee/sharemind/

[7] M. Rabin, "Efficient dispersal of information for security, load balancing, and fault tolerance," *Journal of the ACM (JACM)*, vol. 36, no. 2, pp. 335–348, 1989.

[8] R. Dingledine, N. Mathewson, and P. Syverson, "Tor: The second-generation onion router," in *Proceedings of the 13th conference on USENIX Security Symposium-Volume 13*. USENIX Association, 2004, pp. 21–21.

[9] G. Zhao, M. G. Jaatun, A. Vasilakos, Å. A. Nyre, S. Alapnes, Q. Ye, and Y. Tang, "Deliverance from Trust through a Redundant Array of Independent Net-storages in Cloud Computing," in *Proceedings of IEEE Infocom*, 2011.

[10] M. Abadi and R. Needham, "Prudent engineering practice for cryptographic protocols," *Software Engineering, IEEE Transactions on*, vol. 22, no. 1, pp. 6–15, 1996.

[11] N. E. Larsen, "Privacy in the polippix project," in *D 7.3 PRISE Conference Proceedings: "Towards privacy enhancing security technologies - the next steps"*, 2009, pp. 143–149.

[12] R. Rivest, "Chaffing and winnowing: Confidentiality without encryption," *CryptoBytes (RSA laboratories)*, vol. 4, no. 1, pp. 12–17, 1998.

[13] Google, "Google Docs - Online documents, spreadsheets, presentations, surveys, file storage and more," 2011, http://docs.google.com.

[14] D. Chaum, "The dining cryptographers problem: Unconditional sender and recipient untraceability," *Journal of Cryptology*, vol. 1, pp. 65–75, 1988, 10.1007/BF00206326. [Online]. Available: http://dx.doi.org/10.1007/BF00206326