An efficient secure shared storage service with fault and investigative disruption tolerance

Stelios Erotokritou
Computer Science Department
University College London
London, UK
S.Erotokritou@cs.ucl.ac.uk

Srijith K. Nair
Security Futures Practice, BT
Orion Building, Adastral Park
Ipswich, UK
srijith.nair@bt.com

Theo Dimitrakos
Security Futures Practice, BT
Orion Building, Adastral Park
Ipswich, UK
theo.dimitrakos@bt.com

Abstract – **In this work we focus on solutions to an emerging threat to cloud-based services – namely that of data seizures within a shared multiple customer architecture. We focus on the problem of securing distributed data storage in a cloud computing environment by designing a specialized multi-tenant data-storage architecture. The architecture we present not only provides high degrees of availability and confidentiality of customer data but is also able to offer these properties even after seizures of various parts of the infrastructure have been carried out through a judicial process. Our solution uses a novel way of storing customer data ─ combining the cryptographic scheme of secret sharing and combinatorial design theory, to ensure that the requirements of the architecture are met. Furthermore, we show that our proposed solution is efficient with respect to the amount of hardware infrastructure required, thus making the implementation and use of our proposed architecture cost-efficient for adoption by IT enterprises.**

Keywords: Cloud Storage, Secure Storage, Data Security, Data Seizure

I. INTRODUCTION

Recent advances in networking and on-demand utility based usage models for IT services have seen the emergence of the outsourcing model - where IT capabilities of institutions are outsourced to external parties. This could range from specific processes and services (like CRM) on a Software as a Service (SaaS) platform, to direct hosting of an enterprise's IT infrastructure on an Infrastructure as a Service (IaaS) platform. This has several advantages, including allowing enterprises to operate with lower capital expenditure and freeing up of resources for them to concentrate on their core business.

In all these models, it is essential for the service provider to ensure that customer data hosted on its infrastructure are secure and readily available as and when required by the customer. Such requirements are important when data storage is the provider's main service, like in Amazon S3 [8] or when storage is part of the overall service provided - as in Amazon EC2 [1]. The provider needs to ensure not only that a specific customer's data confidentiality and integrity is protected, but also that it is able to provide business continuity to customers by ensuring that the data can be accessed by customers - even in the presence of hardware and other related failures.

A core tenant of providing a software or infrastructure as a service model for IT resources is that of multi-tenancy, in which services belonging to multiple customers are hosted on the same physical servers. This requires that the resources be shared across these multiple customers. The same is also the case with data storage.

Even though the security of data storage is a very important upcoming issue, there does not seem to be much work in the literature that considers data storage and data escrow and how these two can co-exist together in industry.

The authors are of the opinion that these are emerging threats to cloud-based services and a deterrent to their widespread use. In this paper we consider a new security model for services with the aim of ensuring confidentiality and availability of distributed data storage. This will allow for a robust network architecture for both customers and service providers against common threats to confidentiality and availability as well as the threat of service disruption through escrow.

As motivation for this work, consider the following scenario. A cloud based storage company is offering its service to the public. In order to achieve economies of scale, the data of customer *A* is hosted on shared hardware along with the data of several other customers. As an example, we assume that one such customer, *X,* is engaged in questionable activities and is under investigation by a government authority. As part of the investigative process, it may be the case that *X*'s data will have to be seized. If this occurs, the service provider will be forced to hand over storage disks in order for authorities to gain access to *X*'s data. However, in the process they will be forced to hand over data belonging to multiple other customers. This means that without other preventive measures, not only is *A*'s data unavailable to *A* when needed but also that *A*'s data has now fallen into the hands of a third party, threatening its confidentiality and integrity, through no fault of *A*.

Simple solutions to the problem, such as storing each customer's data on a separate disk drive and/or having multiple copies of the data, are either prone to unavailability in the face of hardware failures or are inefficient regarding resource usage and economies of scale.

In this work we focus on this specific problem and present a multi-tenant data storage architecture geared towards storage service providers. The architecture not only provides high degrees of availability and confidentiality of customer data in the default setup but is also able to offer

these properties to customers in the face of hardware failures or even after parts of the infrastructure have been seized through a judicial process. Our solution uses a novel way of storing customer data, combining the concepts of secret sharing and combinatorial design theory to ensure that the requirements of the architecture are met. Furthermore, we show that our proposed solution is efficient with regards to the amount of infrastructure required, thus making implementation and use of our proposed architecture cost effective for use by an IT enterprise providing the service.

It should be pointed out that this paper presents the architecture design and carries out a theoretical analysis of its requirements. Implementation details of the proposed architecture are considered beyond the scope of this paper and will be carried out along with experimentation and evaluation of the architecture in future work. .

The rest of the paper is structured as follows. In Section 2 the problem statement is defined in more detail, laying down the system and threat models among others. Section 3 presents the proposed solution taking into account our primary threat model and Section 4 provides the security proof of the proposed solution. Section 5 considers our proposed solution against our secondary threat model, while Section 6 compares our proposed solution to other possible solutions. Sections 7 and 8 outline extra properties which could be included to our architecture to provide additional properties while Section 9 looks at related work in this area. We conclude in Section 10 with a statement of future work that could be carried out.

II. PROBLEM STATEMENT

A. System Model

A simplified architecture outline of a multi-tenant storage service for customer data is given in Figure 1.

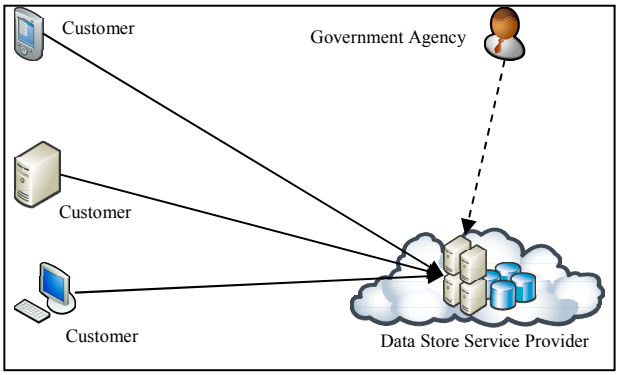

Figure 1. Architecture of model for data storage with entities involved.

Three main entities can be identified in the architecture:

- Data Store Service Provider (DSSP) – A DSSP is a service provider that owns the infrastructure and has the expertise to provide large scale of (secure) storage and the management of data belonging to external customers. This is provided as a service which customers can rent either on long terms contracts or using the cloud computing based pay-as-you-use utility pricing models. The DSSP could either expose the data service directly to the consumer (like Amazon S3) or use it to power other services of which data storage is an integral part (like Amazon EC2).
- Customers - Customers can be individuals or organizations that hire the service of DSSPs for the secure storage and management of their data.
- Government Agency (GA) – A GA can be any agency which using a court order may intervene in and disrupt the service provided by a DSSP. A GA may intervene by demanding – through legal means, access to the data of a customer of a DSSP stored in data center of the DSSP.

B. Threat Model

The threat model associated with the data storage service we consider in this paper is two-fold.

The first is the threat to the service due to hardware failure. It could happen that a part of DSSP's storage infrastructure suffers an outage. This could happen due to any number of reasons such as the failure of the actual disk hardware, electricity failure of the datacenter or even a denial of service (DoS) attack against the service infrastructure. While a complete and total collapse of the storage infrastructure is hard to mitigate against, any proposed framework should be able to cope with a partial failure of the infrastructure.

The second threat associated with the storage service is that of seizure of part of the storage infrastructure by the GA during the course of a court-approved investigation. Again, the objective is that such an action will not render the whole infrastructure of the DSSP out of service. The difference between this threat scenario and the first is that in this one, in addition to the inability to provide service to customers whose data reside in the seized hardware, the confidentiality of the data is also a concern for those customers who are not under investigation but whose data happens to be stored on hardware seized by the authorities. While the hardware is seized so that authorities can have access to the data of the customer under investigation, it is of outmost importance that they should only have access to the data of the specific customer and cannot breach the confidentiality of data belonging to other customers.

In the above threat models, we assume that the DSSP is a trusted service provider and that its customers have full confidence in the secure management of their data. Another model we will also consider in this paper is when this is not true, i.e. in addition to investigative intervention, customers do not trust their DSSP either. We outline the extra steps that need to be taken so that the security of customer data in this stronger security model is equivalent to that of the first.

C. System Requirements

The main requirement of the architecture will be to provide a high degree of availability of customer data. The architecture should therefore be able to service customers uninterruptedly even when a high fraction - up to 50%, of data stores are unavailable. Additionally, as the architecture

is a secure data storage service, it should also provide customer data with high degrees of data confidentiality.

Taking into account our second threat model, the architecture should fulfill the above two requirements even when the data of a single customer is seized by a government authority. Put differently, despite the seizure of system resources by a government authority (to obtain the data of a specific customer), the data of other customers should still be available to them. Additionally, the government authorities should not be able to compromise the confidentiality of the data of the other customers from the seized resources.

III. PROPOSED SOLUTION

In this section we present the proposed storage architecture designed to withstand the threat scenarios described in Section 2. We start with some basic preliminaries as background to the solution.

A. Preliminaries

1) Secret sharing

A t-out-of-n threshold secret sharing scheme allows for a secret M to be split up into a selection $\{s_1,\ldots, s_n\}$ of shares so that the following properties are achieved:

- Any collection of t number of shares is able to reconstruct the secret M.
- Given any subset of (t-1) or less number of shares, no information can be obtained about the secret M.

We will be using Shamir secret sharing scheme as outlined in [11] for our secret sharing purposes.

2) t-designs

A t-(v, k, λ) design [4] is a pair (X, B) where X is a v-set of *points* and B is a collection of k-subsets of X (*blocks*) with the property that every t-subset of X is contained in exactly λ blocks.

B. General Architecture

In order to enable the architecture to offer a high degree of availability and confidentiality for the storage of customer data, the cryptographic scheme of secret sharing will be used.

The data of all customers will be secret-shared using a $t+1$-out-of-$2t+1$ secret sharing scheme. This method of secret sharing data provides a high degree of availability as it can withstand the unavailability of up to t data stores which store shares of customer data (as shown later) whilst maintaining the availability and confidentiality of data.

Because of this, the system architecture can ensure availability of customer data with a high probability – even when up to t data stores which store customer data shares under normal circumstances become unavailable.

Figure 2 below shows how the resilience level of the architecture changes as the number of data stores which become unavailable increases. It shows the number of failures that can be tolerated until customer data cannot be reconstructed. For simplicity we have assumed the value of $t=7$.

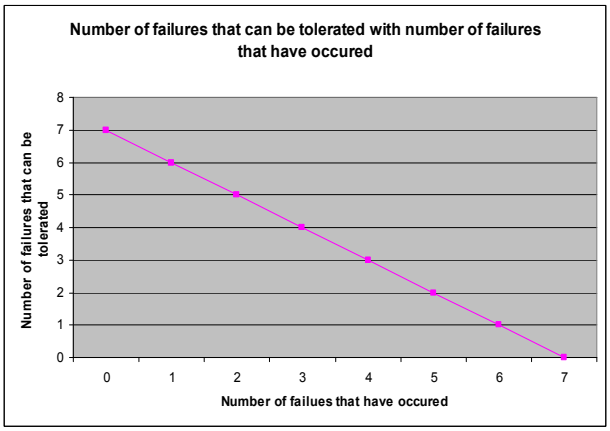

Figure 2. Failures that can be tolerated as the number of failures that have occurred increases

Secret sharing also provides confidentiality of customer data as it ensures that the data is not stored in a single data store. For unauthorised access of customer data to occur, one has to breach the security safeguard put in place by gaining access to at least $t+1$ data stores. If someone was to breach the security of t or less data stores, the secret sharing scheme used in the design ensures that no information about the customer data is recovered.

The architecture for data storage consists of $2t+2$ data stores that enables it to accommodate the storage of data for a maximum of $t+1$ customers. The available $2t+2$ data stores are then classified into two different sets. The way shares of data are stored in the data stores is shown in Figure 3 below.

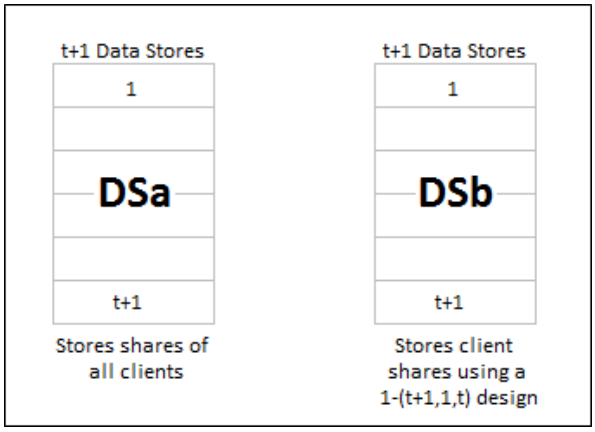

Figure 3. Storing shares of customer data over the data stores

In the first of the two sets of data stores (**DSa**) shares from all customers will be stored. In the second set of data stores (**DSb**), shares of customers will be stored using a 1-($t+1$, 1, t) design as will be explained below. It should be noted that despite the data stores shown to be grouped together in Figure 3, one could follow best practise and house them in different physical locations.

C. Actions Taken When a New Customer Joins

When a new customer joins, the first step in the process of data storage is to carry out secret sharing of the customer's data.

The secret sharing process will create $2t+1$ shares of the customer data. $t+1$ of these shares will be stored in the $t+1$ data stores of **DSa** – storing only one share per data store and storing each share only once.

The remaining t shares of customer i data will be stored in the data stores of **DSb** in a similar way – storing each share only once and one share per data store. The difference in the storage of shares between **DSb** and **DSa** is that the shares stored in **DSb** will be stored in such a way so that for customer $1 \leq i \leq t+1$, the i^{th} data store of **DSb** will not store any shares of customer i data, i.e. one of the data stores of **DSb** will not store any of customer i shares. This particular way of storing customer data is equivalent to a 1-($t+1$, 1, t) design as we ensure that shares of a particular customer are present in t of the $t+1$ data stores of **DSb**.

Consider a setup which serves 4 customers (i.e. $t = 3$). Each customer's data would have been split into 7 (= $2t+1$) shares. Four shares of each dataset are stored over the four datasets in **DSa**. The remaining shares are stores in **DSb** data stores. After the fourth customer has been added to the system, the state of the data stores will be as shown in Figure 4 below.

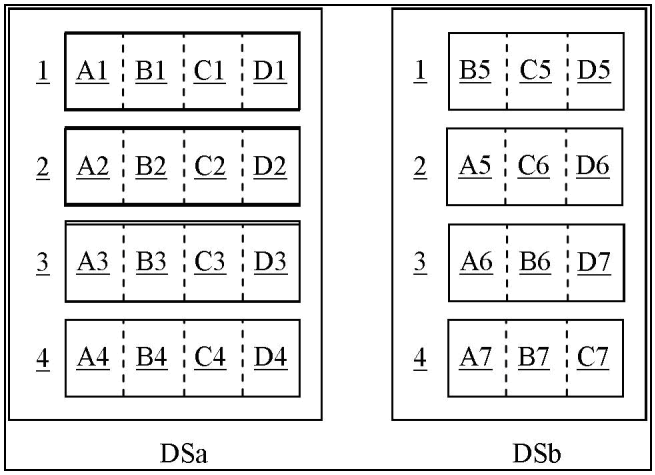

Figure 4. Example state of architecture with the data of four custormers

The first store in **DSa** stores the first share A1 of dataset DA, the first share B1 of dataset DB, the first share C1 of dataset DC, and the first share D1 of dataset DD. The second store has shares A2, B2, C2, D2, the third store has shares A3, B3, C3, D3, and the fourth store has shares A4, B4, C4 and D4. The remaining $t=3$ shares of the first dataset DA are stored across stores of **DSb** - storing only one share per store and storing each share only once. Accordingly, all but one of the stores in **DSb** stores one of the shares of the first dataset. This process is generally repeated for each dataset i to be stored, where $1 \leq i \leq t+1$, and the i^{th} store of **DSb** will not store any shares of user i data.

D. Actions Taken When Customer Data is Updated

When customer data is updated, the new modified data will have to be secret shared. The shares produced are then stored in the corresponding data stores thus deleting the original shares of the data originally stored.

E. Actions Taken When a Customer Leaves

When a customer leaves, all that needs to be done is to delete all shares of the customer data from all the respective data stores where customer shares can be found – all **DSa** data stores and all but one data stores in **DSb**. The extra slot from this customer departure can then be made available to another customer.

F. Actions Taken When More Customers than can be Accommodated Enter System

As stated earlier, the architecture can accommodate the data of only $t+1$ customers. When the $(t+2)^{nd}$ customer wants to store their data in this service, the customer data cannot be stored in the existing setup of the architecture. To accommodate this new customer, a new setup of the architecture will have to be implemented. This new implementation is totally independent of the first implementation and can service a further $t+1$ customers.

G. Actions Taken Upon Customer Data Government Seizures

When government authorities request access to customer i's data for legal reasons, $t+1$ shares of the customer data will need to be made available to them. In this way, the government authorities will have enough shares to reconstruct the customer data. The $t+1$ shares of the customer data handed to the authorities will be the following:

The t shares stored in **DSb** – the authorities will seize all data stores in **DSb** except data store i.

One of the shares in **DSa** – for example the i^{th} data store in **DSa** will be seized by authorities.

As an example, if the data DA of customer A, is requested, in order to reconstruct DA (comprising shares A1-A7), one requires seizure of stores 2 to 4 in **DSb** (providing the three shares A5, A6 and A7) and any one of the stores from **DSa** (providing a fourth share from any one of shares A1-A4).

IV. Security of Architecture against Government Seizures

We now show how the method of storing shares in data stores and how the specific way of handing over data stores to authorities allows for the architecture to continue servicing other customers. We also show how this prevents government authorities from breaching the confidentiality of data for other customers (beyond the customer whose data was seized). In this way, the architecture is secure against any government seizures which may occur.

A. Availability of Customer Data Upon Government Seizures

As stated earlier, when a government seizure of customer *i* data occurs, only one data store of **DSa** will be provided to the authorities. All data stores in **DSa** hold shares of all customer data. As a result of this, after a customer seizure, the architecture always has *t* shares of all customer data – the *t* shares from the data stores which remain in **DSa**.

From **DSb**, all but the i^{th} data store – *t* in total, will be handed to authorities. The i^{th} data store does not hold any of customer *i* data shares. But it does store shares of all other customer data.

Overall, after a government seizure *t+1* shares of data for all other customers remain in the architecture and thus the service can guarantee the availability of customer data.

B. Confidentiality of Customer Data Upon Government Seizures

We now show that despite the data stores seized by government authorities for customer *i* data, the authorities do not hold enough shares for all other customer data – thus maintaining the confidentiality of all other customer data.

As stated earlier, when GA seizes data of customer *i*, only one data store of **DSa** will be provided to the authorities. All data stores in **DSa** hold shares of all customer data. Authorities thus have access to one share of all customer data. From **DSb**, all but the i^{th} data store – *t* in total, will be handed to authorities. For customer $1 \leq j \leq t+1$ where $j \neq i$ all but one of the *t* data stores handed to authorities store shares of customer *j* data. More precisely, the j^{th} data store will not hold any of customer *j* data shares – this is achieved due to the design used in the storage of data shares in **DSb**. From this, only *t-1* shares of all customer data are obtained from **DSb** data stores.

Continuing with the existing example from Section 3, the combination of stores 2 to 4 in **DSb** only provides two shares of each of the datasets DB, DC and DD, whereby the addition of a store from **DSa** provides only one additional share of each of the datasets DB, DC and DD, amounting to three shares overall, which is below the threshold of shares required to reconstruct any of the datasets DB, DC and DD.

Overall, after any government seizures, only *t* shares of customer *j* data will be available to GA. As the data is secret shared such that *t+1* shares are needed to recover data, government authorities cannot reconstruct the original data with the *t* shares they hold and thus the confidentiality of customer data against government seizures is maintained.

C. Deciding the Value of t

The secret sharing scheme used for the sharing of the customer data decides the value of *t*. Within a design implementation a specific range for different values of *t* can be made available to customers. Customers with the same values of *t* can have their data stored on the same implementation of the architecture.

This raises the question of how a data storage service provider decides the value of *t* to offer to clients. This generally depends on the size of a provider and the size of its customer base.

A large value of *t* will result in implementations with large amounts of unused storage space until the capacity of clients who can be accommodated by the implementation is reached. Such an implementation will generally incur higher running costs until its full capacity is reached – rather than one where the value of *t* is lower.

Despite this, once the provisioning of infrastructure is carried out for implementations with large values of *t*, it can accommodate new customers without any extra provisioning – which will be required for implementations with lower values of *t* when their full capacities are reached earlier. Additionally, once the full capacity of an implementation is reached then when larger values of *t* are used, , lower overall running and setup costs will be incurred than multiple implementations of lower values of *t* which will be required to cover the same number of customers.

It seems that small values of *t* would be more favorable for smaller providers who try to maximize usage capacities and try to keep running costs and unused storage space to a minimum. Large providers on the other hand would be able to bear the cost of unused storage space until the number of clients rises.

Large providers would thus be able to offer a wider range of *t* values to their customers, thus gaining a competitive advantage over smaller providers, who provide lower values of *t*, with associated lower security guarantees.

V. CONFIDENTIALITY WHEN USING NON-TRUSTED DSSP

We now consider the second threat model which states that the storage service should ensure the confidentiality of customer data even when a non-trusted data storage service provider is used. We consider this in addition to the existing threat of government seizure.

Since the shares of data are stored on data stores owned and maintained by a non-trusted DSSP, the DSSP has access to all the shares of the data. Because of this, the DSSP is able to reconstruct the original data, breaking the confidentiality of the data. It is thus important that an extra layer of security is designed into the architecture to preserve the confidentiality of customers' data in this threat model.

This is solved by first encrypting the data to be stored. This could be carried out using encryption algorithms such as AES and 3DES – using key sizes relative to the preferences of the customer. The encryption of the data will need to be carried out by the customers themselves who will then provide the encrypted data to the non-trusted DSSP. In turn, the DSSP carries out secret sharing of the data and stores the shares in the same manner as outlined earlier in the description of the architecture. In this way, the non-trusted DSSP cannot break the confidentiality of the customer data as the data is encrypted.

However, a problem arises when a GA needs access to a customer's data. Should government authorities seize data stores as earlier outlined, they will only be able to reconstruct the encrypted data.

Getting around this problem is a techno-legal process. The technical step that needs to be added to the above setup (to allow government authorities to recover the decrypted format of customer data when required) can be achieved by

making it a legal requirement to carry out a *2*-out-of-*2* secret sharing of the key used in the encryption of the data by the customer. The secret sharing of the key will be carried out by the customers themselves. Customers will also retain the value of the encryption key so that they can decrypt their data.

One of the shares of the key will then have to be given to the DSSP by the customer. As the DSSP holds only one share of the key and two are needed to recover the key, the DSSP cannot learn the encryption key and confidentiality of data is preserved. The other share of the key will be given to the government authorities by the customer.

Upon request by government authorities to a DSSP for customer data, data stores as earlier outlined will be given to them as too will the share of the encryption key held by the DSSP. With this solution, government authorities hold two shares of the encryption key and can thus recover the original unencrypted customer data[1].

VI. ADVANTAGES OF PROPOSED ARCHITECTURE OVER ALTERNATIVE SOLUTIONS

In this section we present other possible architectures that can achieve the system requirements set out earlier and argue why the proposed solution is better.

A. Store Individually Encrypted Customer Data

An alternative way to achieve the system requirements is to store the encrypted data of a customer individually and separately from the data of other customers. Implementation in this manner protects data from an untrusted service provider (as the data is encrypted) and also allows availability of data when the data of customers are seized.

The great disadvantage of this implementation method is that in order to offer a high degree of availability to customer data, the data of the customer will need to be replicated many times. Because of this, many data stores will be required for the storage of the customer data. Contrary to our proposed architecture - where for each customer only two data stores (on average) are required, this alternative implementation will require many more data stores per customer to offer the same levels of availability guarantees as our proposed solution. This raises costs for service providers and in turn for the customers.

It is thus easy to see that the proposed architecture is a more viable cost worthy solution – both for customers and the service provider.

[1] A problem can be identified with the given specific solution. This is that a customer can secret share a key which is different to the encryption key used to encrypt the customer data. Because of this, even if the data are seized, the customer data will not be learned by government authorities – which goes against one of the key properties of the architecture. However, this problem is present in other secure storage services. Legal steps exist which can be taken so that government authorities are able to obtain the encryption key. An alternative solution is one where customers hand their data to a trusted third party which encrypts the data - and then hands this to the data service provider, secret shares the encryption key and hands the appropriate shares to the corresponding entity. The third party also provides customers with the value of the encryption key used. This proposed solution though does suffer in the secure implementation of the trusted third party.

B. Store Encrypted Customer Data Together With Other Customer Data

Another alternative solution is one in which all customer data is stored together in encrypted format in a data source and this data source is replicated an appropriate number of times. This solution is more viable than the previous alternative solution, as less data stores are needed to offer a high degree of availability.

The problem with this alternative solution is that the data of many customers are stored in encrypted form in each data store (which will be replicated over the architecture). This allows for someone to gain access to the encrypted form of the secret data of many customers by gaining access to just one of these data stores.

Although this data will be encrypted with currently secure encryption algorithms, the confidentiality of the data could be broken in the future. This may occur when insecurities of the encryption algorithms used are found or when alternatively, the secrecy of encryption keys is lost (due to bad key management).

When any of the two scenarios occurs, this could lead to the loss of data confidentiality. If the first were to occur this would enable the data of all customers - for which the data were stored in the data store one had access to, to lose their confidentiality. On the contrary, as the proposed architecture uses secret sharing to provide confidentiality of data one would have to gain access to at least *t+1* data stores for them to learn the data[2]. This is more difficult than accessing a single data store.

Also, contrary to using encryption only for the confidentiality of data, if one were to get access to data shares over *t* or less data stores then the confidentiality of the data will never be broken.

Using the proposed architecture presented in this work therefore provides customers with far greater guarantees to the confidentiality of their data.

VII. TOLERATING MORE THAN ONE CUSTOMER SEIZURE

The system architecture presented so far is able to provide availability and confidentiality of customer data when government seizures occur against a single customer in a specific implementation.

If a second customer's data seizure were to occur, our current solution would require for more data stores to be handed over to the authorities in order to provide them with *t+1* shares for the customer data they are seeking.

However, by handing to them this extra data store, the availability of data for the remaining *t-1* customers will be affected because now, as per the current implementation, there would only be *t* data stores and thus only *t* shares of customer data - which are not enough to recover any of the customer data. It will also affect the confidentiality of data for the remaining customers, as government authorities will

[2] The joint use of encryption and secret sharing could also be carried out in our proposed solution - where encrypted data is secret shared and stored over the data stores. This provides a higher degree of confidentiality. For one to break the confidentiality of data, *t+1* data stores will need to be accessed and the data will need to be decrypted also.

now hold *t+1* shares of all customers – but this is easily fixed by employing the solution of encrypting the customer's data before it is sent to the service provider.

A way of decreasing the probability of two customer seizures occurring in the same implementation would be to ensure that data from customers related to each other (for example subsidiaries of the same company) are not stored in the same architecture implementations. Doing this prevents the possibility of one customer's data being seized as a consequence of another customer's data being seized - in case the customers are closely linked or related to each other.

Another way of overcoming this problem is outlined below. It ensures that customer data is always available and also that the confidentiality is also maintained – irrespective of the number of customer seizures that may occur.

To achieve this, a greater number, *3t+3*, of data stores will need to be used to support *t+1* customers – thus averaging three data stores per customer. The way shares of customer data are stored in the data stores is shown in the following Figure 5.

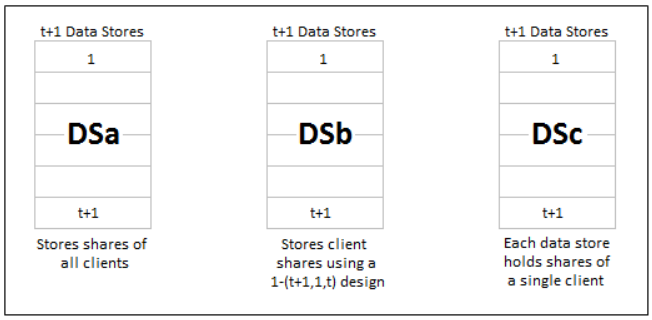

Figure 5. Storing shares of customer data over the data stores

Contrary to the original description of the architecture, this variation will now have three sets of data stores, all of size *t+1* data stores. The secret sharing of customer data will change from a *t+1*-out-of-*2t+1* secret sharing scheme to a *t+1*-out-of-*2t+2* secret sharing. The first *2t+1* shares will be stored in the same way as the original description of the architecture – storing them in the exact same way over **DSa** and **DSb**. The extra share will be stored in the new set **DSc** of data stores – only storing data shares of customer *i* in data store *i* of **DSc**.

Upon the first GA data seizure request, data stores will be handed to them in the same way as in the original description of the architecture. This allows government authorities to have enough shares (*t+1*) to recover the data of the specific customer. In this way government authorities only hold *t* shares of all other customer data, not allowing them to recover the rest of the customers data.

All that needs to be done if any further customer data is requested by the authorities is to provide one more share of the customer data. In this altered architecture this extra share will be obtained by providing the corresponding data store from **DSc**. This ensures that government authorities will have *t+1* data stores of the specific customer. For the remaining customers, only *t* shares are held by the authority, thus protecting the confidentiality of the customers data. Note that if the data store handed to the authorities came from **DSa** or **DSb** this would provide government authorities with t+1 shares for all customer data allowing them to break the confidentiality of data for all customers – which goes against the requirements of the architecture.

With regard to the availability of other customer data, there are always *t* data stores storing customer shares which remain from **DSa** and one data store remains from **DSb**. One data store from **DSc** also remains for customers whose data has not been seized. Because of this, there are always at least *t+1* data stores holding customer data - which ensures availability of customer data.

Even when data for *t* customers are seized, this new architecture still provides the availability of the last remaining customer data (as *t* data stores from **DSa** and one from both **DSb** and **DSc** will remain). This is very important for the integrity of the service[3].

Figure 6 below shows how the resilience level for this version of the architecture changes as the number of customer data seizures increases. It shows the number of failures that can be tolerated until customer data cannot be reconstructed. For simplicity we have assumed the value of *t=7*.

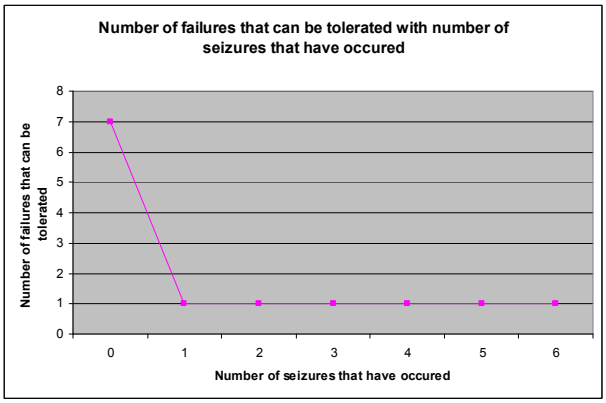

Figure 6. Storing shares of customer data over the data stores

A. *Ensuring Consistency of Data in Seized Infrastructure*

An issue with this design is evident. Suppose that an implementation of the architecture accommodates the data of *t+1* customers. Upon the seizure of data of customer *i*, data stores as specified in the definition of the architecture will be handed to the authorities. Now suppose that after a period of time, the data of customer *j≠i* are also seized. The data store

[3] This version of the architecture which defends against any number of customer seizures can be solved using *3t+1* data stores. However if *t* customer seizures occur *t+1* data stores will remain in the architecture. Although this still allows for availability of data for the last customer, if one of the remaining data stores becomes unavailable (due to a power failure for example), then this would make the data of the last customer unavailable. With the given solution, with t customer seizures *t+2* data stores storing shares of the last remaining customer data are available. This provides a higher degree of availability for the last remaining customer – as this can tolerate at least one data store unavailability.

as defined in the specification of the architecture will be handed to the authorities to accommodate this extra seizure. If the data of customer *j* was not altered in the time between the seizures of customer *i* data and the seizures of customer *j* data, then there is no problem. If however the data of customer *j* was altered in that time period, then the shares stored in data stores held by government authorities will not be up to date with the alterations of the data of customer *j*.

To ensure consistency, it is important for the secret sharing of the altered customer data - whilst infrastructure of an implementation is under government seizure, to be consistent with the data shares stored on infrastructure held by government authorities. This is possible as a *t+1*-out-of-*2t+2* secret sharing scheme is used and also because only *t* shares for customer *j* data are held by government authorities. Because of this, secret sharing of altered data can be made consistent with the data held in seized data stores. This is done by exploiting a property of polynomials of degree *t*. A *t* degree polynomial is used to carry out the *t+1*-out-of-*2t+2* secret sharing. For such polynomials, there are *q* (where *q* denotes the size of the finite field used in the secret sharing scheme) polynomials which share the same *t* points. In this case the *t* shares which can be shared over the alterations will be those stored in the data stores seized by authorities. This allows for the consistent and secure update of customer data - even when shares stored in seized data stores retain the same values.

B. *Adding Extra Availability after Customer Seizures*

The current solution to the architecture (*t+1*-out-of-*2t+2*) is able to ensure continuity of customer data after a government seizure occurs. It also provides high degrees of availability. Upon a government seizure - due to the taking away of infrastructure, availability guarantees decrease. In its current form the architecture is able, after a government seizure, to cope with the non-availability of at most one data store from **DSa** and **DSb** (one overall) or up to t data stores from **DSc** and still allow for the data of all customers to be available to them. This degree of availability guarantee is evidently much lower compared to when there are no authority seizures.

However, the availability of the architecture could be increased further by creating new shares (using Lagrange interpolation) from the shares which remain in the architecture. The number of new shares which will need to be created depends on the availability guarantees agreed with the customer and the service provider as part of a service level agreement.

VIII. Additional Architecture Properties

In this section we consider ways to provide additional properties which could be included to improve the proposed architecture, with brief descriptions of how these can be achieved.

A. *Storing Previous and Updated Values Until Commit*

In case customer data needs to be altered but not immediately committed to storage, the design of the architecture could easily accommodate this. The architecture would need a separate data storage entity which would store the difference between the old and the updated value of the data. When it is time to commit the updated data, the old data will need to be reconstructed from the data shares in the architecture data stores. The updated value will then need to be calculated using the reconstructed old value and the stored difference. Finally the updated value will need to be secret shared and these new data shares will need to overwrite the previous data shares of the old value[4].

B. *Proving Integrity of Data Shares to Customers*

In case customers need to check the integrity of their stored data, this could be provided by the architecture using hash functions (such as SHA-1). Upon creation of data shares, hashing of these shares could be carried out. The hash values are in turn provided to customers. Whenever customers want to check the integrity (and availability) of their data, customers can query the architecture which will locate the customer data shares and calculate their hash values. These will then be sent back to the customer who can compare the original hash values (provided to the customer when data are primarily stored) to the received hash values. In this way, customers can evaluate the integrity (and availability) of their stored data.

It should be noted that this can be an automated process running on the customer systems and serviced by the service provider architecture.

If it is found that the integrity of shares falls below an acceptable level (for example if a third of the customer data shares are corrupt) then appropriate steps can be taken to correct any errors that may have been found. In this case the correct values of error data shares can be recalculated - using Lagrange interpolation, and thus an appropriate number of correct shares can be maintained.

IX. Related Work

Cryptographic secret sharing schemes [11] have been used to distribute a secret among participants in such a way that each participant gets a share of the secret and the secret can only be reconstructed when a specified sufficient number of these individual participant shares are combined together. The primitive has however not been used in providing service continuity for multiple customers in the face of unavailability of parts of the storage infrastructure.

Amazon Web Services is one of the major cloud providers to provide data storage services at an industrial level. However, the service does not provide any form of confidentiality and integrity guarantees. Certain storage providers cater for only specific type of data such as Nirvanix [7] which focus on the optimization of storage for media files. The importance for the security and availability of data stored in such providers has been addressed by different papers [2, 3, 5, 10, 12].

Additionally to providing security to the data stored in such services, extra requirements have been considered so

[4] An alternative and more efficient solution is to add the difference between the two values (old and altered) to all stored data shares.

that they can be offered by such services. The verification of data whether by the service itself or remotely by customers is one such service and this has been looked at in previous work such as [6, 9, 13]. The work presented in [13] concentrates on ensuring the correctness of users' data in a remote 'cloud'-based infrastructure. It can also be used to pinpoint the servers among which the distributed data is misbehaving. While [3] in addition supports dynamic operations like update, delete and append on the data, it does not tackle the central issue addressed in this work - of performing the cryptographic computation and data piece distribution in such a way so as to ensure secure tolerance to missing/offline/seized data storage parts.

The work reported in [14] again concentrates on providing publicly verifiable secure data storage while [15] looks at a form of threshold secret sharing scheme that supports verifiability as well as the ability to dynamically add or remove shareholders. Neither of them considers the problem that forms the core of our work nor can their solutions be used to address the issues solved by our designs.

Even though the security of data storage is a very important upcoming issue, there does not seem to be much work in the literature which considers data storage and data escrow and how these two can co-exist together in industry. In this work this has been considered along with related work aspects to data storage that were mentioned. As a result of this we consider the work we present as an important step to making data storage in industry feasible with minimal operational and legal issues.

X. Conclusions and Future work

In this work we have focused on the security and availability of customer data for a data store service provider under specific circumstances where the escrow of customer data by government authorities may take place. We have proposed a data storage architecture for this scenario. We believe our work to be of importance to data storage services as it is able to achieve this with an efficient and elegant system architecture that achieves the requirements with a fairly low demand for infrastructure – leading to lower costs and thus great industrial competitive advantages.

As future work, a more complete analysis of our proposed architecture will be carried out. This will include an experimental evaluation of an implementation of the presented architecture.

As a continuation to our work, it will be interesting to see how future work can refine our proposed architecture to deal with a more authoritarian government intervention which may demand all infrastructures where the data of a customer may lie.

References

[1] Amazon Inc. Amazon Elastic Compute Cloud (Amazon EC2), Dec 2008. [Online] Available: http://aws.amazon.com/ec2/.

[2] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson and D. Song, "Provable Data Possession at Untrusted Stores," Proc. of CCS '07, pp. 598–609, 2007.

[3] K. D. Bowers, A. Juels, and A. Oprea, "HAIL: A High-Availability and Integrity Layer for Cloud Storage," Cryptology ePrint Archive, Report 2008/489, 2008, http://eprint.iacr.org/.

[4] C.J. Colbourn and J.H. Dinitz, The CRC Handbook of Combinatorial Designs, CRC Press, Inc., New York, 1996.

[5] D. L. G. Filho and P. S. L. M. Barreto, "Demonstrating Data Possession and Uncheatable Data Transfer," Cryptology ePrint Archive, Report 2006/150, 2006, http://eprint.iacr.org/.

[6] J. Hendricks, G. Ganger, and M. Reiter, "Verifying Distributed Erasurecoded Data," Proc. 26th ACM Symposium on Principles of Distributed Computing, pp. 139 – 146, 2007.

[7] "Nirvanix," http://www.nirvanix.com.

[8] M. Palankar, A. Onibokun, et al., "Amazon S3 for Science Grids: a Viable Solution," in 4th USENIX Symposium on Networked Systems Design & Implementation (NSDI'07), 2007.

[9] T. S. J. Schwarz and E. L. Miller, "Store, Forget, and Check: Using Algebraic Signatures to Check Remotely Administered Storage," Proc. of ICDCS '06, pp. 12–12, 2006.

[10] M. A. Shah, M. Baker, J. C. Mogul, and R. Swaminathan, "Auditing to keep online storage services honest," in Proc. of HotOS'07. Berkeley, CA, USA: USENIX Association, 2007, pp. 1–6.

[11] A. Shamir, "How to Share a Secret", Communications of the ACM, 22(11): 612-613, 1979.

[12] C. Wang, Q. Wang, K. Ren, and W. Lou, "Ensuring Data Storage Security in Cloud Computing," in Proc. of IWQoS'09, Charleston, SC, USA, July 2009.

[13] Q. Wang, C. Wang, J. Li, K. Ren, and W. Lou, "Enabling public verifiability and data dynamics for storage security in cloud computing," in Proc. of ESORICS'09, Saint Malo, France, Sep. 2009.

[14] J. Yu, F. Kong, R. Hao and Z. Cheng, "A Publicly Verifiable Dynamic Sharing Protocol for Data Secure Storage", pp.471-472, 2008 The Ninth International Conference on Web-Age Information Management, 2008.

[15] T. M. Wong, C. Wang and J. M. Wing, "Verifiable Secret Redistribution for Archive Systems", Proceedings of the First International IEEE Security in Storage Workshop, pp. 94-105, 2002.