SECURING CLOUD DATA UNDER KEY EXPOSURE

[1]Dr U Feroze Khan, [2]Dr K G S Venkatesan, [3]Dr Nedunchezhian, [4]Dr Prabakaran
1,2,3,4 Professor, Department of Computer Science and Engineering,
Malla Reddy College of Engineering, Hyderabad

Abstract—Recent news reveal a powerful attacker which breaks data confidentiality by acquiring cryptographic keys, by means of coercion or backdoors in cryptographic software. Once the encryption key is exposed, the only viable measure to preserve data confidentiality is to limit the attacker's access to the cipher text. This may be achieved, for example, by spreading ciphertextblocksacrossserversinmultipleadministrativedomains—thusassumingthattheadversarycannotcompromiseall of them. Nevertheless, if data is encrypted with existing schemes, an adversary equipped with the encryption key, can still compromise a single server and decrypt the cipher text blocks stored therein. In this paper, we study data confidentiality against an adversary which knows the encryption key and has access to a large fraction of the cipher text blocks. To this end, we propose Bastion, a novel and efficient scheme that guarantees data confidentiality even if the encryption key is leaked and the adversary has access to almost all cipher text blocks. We analyze the security of Bastion, and we evaluate its performance by means of a prototype implementation. We also discuss practical insights with respect to the integration of Bastionincommercialdispersedstoragesystems.OurevaluationresultssuggestthatBastioniswell-suitedforintegrationin existing systems since it incurs less than 5% over head compared to existing semantically secure encryption modes.

Index Terms—Key exposure, data confidentiality, dispersed storage

1 INTRODUCTION

HE world recently witnessed a massive surveillance program aimed at breaking users' privacy. Perpetrators were not hindered by the various security measures deployed within the targeted services [31].For instance, although these services relied on encryption mechanisms to guarantee data confidentiality, the necessary keying material was acquired by means of backdoors, bribe, or coercion.

If the encryption key is exposed, the only viable means to guarantee confidentiality is to limit the adversary's access to the cipher text, e.g., by spreading it across multiple administrative domains, in the hope that the adversary cannot compromise all of them. However, even if the data is encrypted and dispersed across different administrative domains, an adversary equipped with the appropriate keying material can compromise a server in one domain and decrypt cipher- text blocks stored there in.

In this paper, we study data confidentiality against an adversary which knows the encryption key and has access to a large fraction of the ciphertext blocks. The adversary can acquire the key either by exploiting flaws or backdoors in the key-generation software [31], or by compromising the devices that store the keys (e.g., at the user-side or in the cloud). As far as we are aware ,this adversary invalid a tes the security of most cryptographic solutions, including those that protect encryption keys by means of secret-sharing (since these keys can be leaked as soon as they are generated).

To counter such an adversary, we propose Bastion, a novel and efficient scheme which ensures that plaintext data cannot be recovered as long as the adversary has access to at most all but two cipher text blocks, even when the encryption key is exposed. Bastion achieves this by combining the use of standard en- crypt ion functions with an efficient linear transform. In this sense, Bastion shares similarities with the no- tion of all-or-nothing transform. An AONT is not an encryption by itself, but can be used as a pre-processing step before encrypting the data with a block cipher. This encryption paradigm—called AON encryption— was mainly intended to slow down brute-force attacks on the encryption key. However, AON encryption can also preserve data confidentiality in case the encryption key is exposed, as long as the adversary has access to at most all but one ciphertext blocks. Existing AON encryption schemes, however, require at least two rounds of block cipher encryptions on the data: one pre-processing round to create the AONT, followed by an- other round for the actual encryption. Notice that these rounds are sequential, and cannot be parallelized. This results in considerable—often unacceptable—overhead to encrypt and decrypt large files. On the other hand, Bastion requires only one round of encryption—which makes it well-suited to be integrated in existing dispersed storage systems. We evaluate the performance of Bastion in comparison with a number of existing encryption schemes. Our results show that Bastion only incurs a negligible per- show that Bastion only incurs a negligible performance deterioration (less than 5%) when compared to symmetric encryption schemes, and considerably improves the performance of existing AON encryption schemes [12], [26]. We also discuss practical insights with respect to the possible integration of Bastion in commercial dispersed storage systems. Our contributions in this paper can be summarized as follows:

- We propose Bastion, an efficient scheme which ensures data confidentiality against an adversary that knows the encryption key and has access to a large fraction of the ciphertext blocks.
- We analyze the security of Bastion, and we show that it prevents leakage of any plaintext block as long as the adversary has access to the encryption key and to all but two cipher text blocks.

We evaluate the performance of Bastion analytically and empirically in comparison to a number of existing encryption techniques. Our results show that Bastion considerably improves (by more than 50%) the performance of existing AON encryption schemes, and only incurs a negligible overhead when compared to existing semantically secure encryption modes (e.g., the CTR encryption mode).

We discuss practical insights with respect to the deployment of Bastion within existing storage systems, such as the HYDRA stor grid storage system [13],[23].

The remainder of the paper is organized as follows. InSection2,we define our notation and building blocks. In Section 4, we describe our model and introduce our scheme, Bastion. In Section 5, we analyze our scheme in comparison with a number of existing encryption primitives. In Section 6, we implement and evaluate the performance of Bastion in realistic settings; we also discuss practical insights with respect to the integration of Bastion within existing dispersed storage systems. In Section 7, we overview related work in the area, and we conclude the paper in Section 8.

II. PRELIMINARIES

We adapt the notation of[12]for our settings. We define a block cipher as a map $F : \{0, 1\}k \times\{0, 1\}l \rightarrow \{0, 1\}l$, for positive k and l. If Pl is the space of all $(2^l)!$ l-bits permutations, then for any $a \in\{0,1\}k$, we have $F(a,\cdot)\in Pl$. We also write $Fa(x)$to denote $F(a,x)$. We model F as an ideal block cipher, i.e., a block cipher picked at random from $BC(k,l)$,where $BC(k,l)$ is the space of all block ciphers with parameters k and l. For a given block cipher $F \in BC(k, l)$, we denote$F-1\in BC(k,l)$as$F-1(a,y)$oras$F-a1(y)$,for $a \in \{0, 1\}^k$.

Encryption modes

An encryption mode based on a block cipher F/F −1 is given by a triple to f algorithms = (K, E, D)where:

K The key generation algorithm is a probabilistic algorithm which takes as input a security parameter k and outputs a key a ∈ {0, 1}k that specifies Fa and F −1.

E The encryption algorithm is a probabilistic algorithm which takes as input a message x∈{0,1}∗,andusesFaandFa−1asoracles to output ciphertext y.

The decryption algorithm is a deterministic algorithm which takes as input a ciphertext y, and uses Fa and F −1as oracles to output

Plain textx∈{0,1}∗,or⊥ify is invalid.

For correctness, we require that for any key a ← K(1k),−1foranymessagex∈{0,1−1}∗,and for any y←E a a (x), we have x ← Da a (y).

Security is defined through the following chosen- plaintext attack (CPA) game adapted for block ciphers:

```
$\mathbf{Exp}^{ind}(A,b)$
    $F \leftarrow BC(k, l)$
    $a \leftarrow K(1^k)$
    $x_0, x_1, state \leftarrow A^{E^{F_a, F_a^{-1}}}(find)$
    $y_b \leftarrow E^{F_a F_a^{-1}}(x_b)$
    $b' \leftarrow A(guess, y_b, state)$
```

In the in d experiment, the adversary has unrestrictedoracleaccesstoEFa,aF−1duringthe“find”stage.Atthis

point, A outputs two messages of equal length x0, x1, and some state information that are passed as input when the adversary is initialized for the “guess” stage (e.g.,statecancontainthetwomessagesx0,x1).During the “guess” stage, the adversary is given the ciphertext of one message out of x0, x1 and must guess which message was actually encrypted. The advantage of the adversary in the ind experiment is: Advind(A) = |Pr[Expind(A, 0) = 1]−Pr[Expind(A, 1) = 1]

Definition 1. An encryption mode = (K, E , D) is ind secure if for any probabilistic polynomial time (p.p.t.)adversary A, we have AdvQind(A)≤ǫ, where ǫ is a negligible function in the security parameter.

REMARK 1. The ind experiment allows the adversary to see the entire (challenge) ciphertext. In a sce- nario where ciphertext blocks are dispersed across a number of storage servers, this means that the ind- adversary can compromise all storage servers and fetch the data stored there in.

REMARK 2. In the ind experiment (and in other experiments used in this paper), we adopt the Shannon Model of a block cipher that, in practice, instantiates an independent random permutation for every different key. This model has been used in previous related work [3], [12], [17] to disregard the algebraic or cryptanalysis specific to block ciphers and treat them as a black-box transformation.

All or Nothing Transforms

An All or Nothing Transform (AONT) is an efficiently computable transform that maps sequences of input blocks to sequences of output blocks with the following properties: (i) given all output blocks, the transform can be efficiently inverted, and (ii) given all but one of the output blocks, it is infeasible to compute any of the original input blocks. The formal syntax of an AONTi

Q given by a pair of p.p.t. algorithms

E, D) where: = (

E The encoding algorithm is a probabilistic algorithm which takes as input a message x∈

{0,1}∗,and out puts a pseudociphertexty.

D The decoding algorithm is a deterministic algorithm which takes as input a pseudo- cipher text y, and outputs either a message x

∈{0,1}∗or ⊥ to indicate that the input pseudo-cipher text is invalid.

For correctness, we require that for allx∈{0,1}∗, and for all y ← E(x), we have x ← D(y).

The literature comprises a number of security definitions for AONT (e.g., [8], [12], [26]). In this paper, we rely on the definition of [12] which uses the aont experiment below. This definition specifies a block length l such that the pseudocipher text y can be written as y = y[1] . . . y[n], where |y[i]| = l and n ≥1.

```
$\mathbf{Exp}^{aont}(A, b)$
    $x, state \leftarrow A(find)$
    $y_0 \leftarrow E(x)$
    $y_1 \leftarrow \{0, 1\}^{|y_0|}$
    $b' \leftarrow A^{Y_b}(guess, state)$
```

On input j, the oracle Yb returns yb[j] and accepts up to (n − 1) queries. The aont experiment models an adversary which must distinguish between the encod- ing of a message of its choice and a random string (of the same length),while the adversary is allowed access

to all but one encoded blocks. The advantage of A in the aont experiment is given by

$\mathrm{Adv}^{aont}(A) = |\Pr[\mathrm{Exp}^{aont}(A, 0) =1]- \Pr[\mathrm{Exp}_{Q}^{aont}(A,1)=1]|$
Q

Definition 2. An All-or-Nothing Transform = (E, D) isaoQnt secure if for any p.p.t. adversary A, we have Adv (A) ≤ϙ, where ϙ is a negligible function in the security parameter.

Known AONTs

Rivest[26]suggested the package transform which leverages a block cipher F/F −1 and maps m block strings to

n = m + 1 block strings. The first n − 1 output blocks are computed by XORing the i-th plaintext block with FK(i), where K is a random key. The nth output block is computed XORing K with the encryption of each

of the previous output blocks, using a key K0 that is publicly known. That is, given x[1]...x[m],the package transform outputs y[1] . . . y[n], with n = m + 1,where:

$y[i] = x[i] \oplus F_K(i),\ 1 \le i \le n - 1,$

$y[n]=K_{\ i=1} F_{K}0\ (y[i] \oplus i).$

Desai [12] proposed a faster version where the block cipher round which uses K0 is skipped and the last output block is set to y[n] = K in−1 y[i]. Both AONTs are secure according to Definition 2[12].

REMARK 3. Although most proposed AONTs are based on block ciphers [12], [26], an AONT is not an encryption scheme, because there is no secret-key information associated with the transform. Given all the output blocks of the AONT, the input can be recovered without knowledge of any secret.

III. SYSTEM AND SECURITY MODEL

In this section, we start by detailing the system and security models that we consider in the paper. We then argue that existing security definitions do not capture well the assumption of key exposure, and propose a new security definition that captures this notion.

System Model

We consider a multi-cloud storage system which can leverage a number of commodity cloud providers (e.g., Amazon, Google) with the goal of distributing trust across different administrative domains. This "cloud of clouds" model is receiving increasing attention nowa- days [4], [6], [32] with cloud storage providers such as EMC, IBM, and Microsoft, offering products for multi- cloud systems [15], [16], [29].

In particular, we consider a system of s storage servers S1, . . . , Ss, and a collection of users. We assume that each server appropriately authenticates users. For simplicity and without loss of generality, we focus on the read/write storage abstraction of [21] which exports two operations:

write(v)This routine splits v into s pieces
{v1,...,vs}and sends(vj)to server Sj, for
$j \in [1 . . . s]$.

read(•)There ad routine fetches the stored value v from the servers .For each j∈[1...s],piece vj is downloaded from server Sj and all

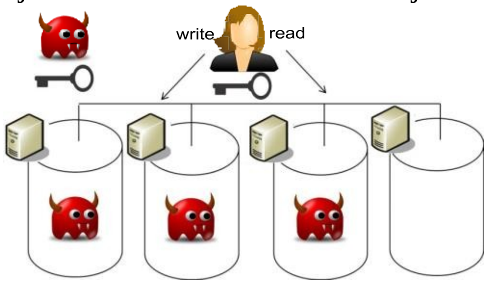


Fig. 1. Our attacker model. We assume an adversary which can acquire all the cryptographic secret material, and can compromise a large fraction (up to all but one) of the storage servers. pieces are combined into v. We assume that the initial value of the storage is a special value ⊥, which is not a valid input value for a write operation.

Adversarial Model

We assume a computationally-bounded adversary A which can acquire the long-term cryptographic keys used to encrypt the data. The adversary may do so either (i) by leveraging flaws or backdoors in the key-generation software [31], or (ii) by compromising the device that stores the keys (in the cloud or at the user). Since ciphertext blocks are distributed across servers hosted within different domains, we assume that the adversary cannot compromise all storage servers (cf. Figure1).

In particular, we assume that the adversary can com- promise all but one of the servers and we model this adversary by giving it access to all but λ ciphertext blocks.

Note that if the adversary also learns the user's credentials to log into the storage servers and downloads all the ciphertext blocks, then no cryptographic mechanism can preserve data confidentiality. We stress that compromising the encryption key does not necessarily imply the compromise of the user's credentials. For example, encryption can occur on a specific-purpose device [10], and the key can be leaked, e.g., by the manufacturer; in this scenario, the user's credentials to access the cloud servers are clearly not compromised.

(n − λ)-CAKE Security

Existing security notions for encryption modes capture data confidentiality against an adversary which does not have the encryption key. That is, if the key is leaked, the confidentiality of data is broken.

In this paper we study an adversary that has access to the encryption key but does not have the entire ciphertext. We therefore propose a new security definition that models our scenario.

As introduced above, we allow the adversary to access an encryption/decryption oracle and to "see" all but λ ciphertext blocks. Since confidentiality with λ = 0 is clearly not achievable1, we instead seek an encryption mode where λ = 1. However, having the flexibility of setting λ ≥ 1 allows the design of more efficient schemes while keeping a high degree of security in practical deployments. (See Remark 7.)

We call our security notion (n−λ) Ciphertext Access under Key Exposure, or (n − λ)CAKE. Similar to [12], (n − λ)CAKE specifies a block length l such that a ciphertexty can be written as y=y[1]...y[n]where
|y[i]| = l and n >1.

```
$\mathbf{Exp}^{(n-\lambda)CAKE}(A, b)$
    $a \leftarrow K(1^k)$
    $x_0, x_1, state \leftarrow A^{E^{F_a, F_a^{-1}}}(find)$
    $y_b \leftarrow E^{F_a F_a^{-1}}(x_b)$
    $b' \leftarrow A^{Y_b, E^{F_a, F_a}}(guess, state)$
```

The adversary has unrestricted access to E a a in both the "find" and "guess" stages. On input j, the oracle Yb returns yb [j] and accepts up to n − λ queries. Ontheonehand,unrestricted oracleaccesstoEaFa,F−1 captures the adversary's knowledge of the secret key. On the other hand, the oracle Yb models the fact that the adversary has access to all but λ ciphertext blocks. This is the case when, for example, each server stores λ ciphertext blocks and the adversary cannot compromise all servers. The advantage of the adversary is defined as:

(n−λ)CAKE (n−λ)CAKE
AdvQ(A)=Pr[ExpQ (A, 1) =1]−
Pr[Exp(Qn−λ)CAKE(A,0)=1]

Definition 3. An encryption mode = (K, E , D) is (n−λ)CAKE secure if for any p.p.t. adversary A, we have AdvQ (A) ≤ ϙ, where ϙ is a negligible function in the security parameter.

Definition 3 resembles Definition 2 but has two fundamental differences. First, (n − λ)CAKE refers to a keyed scheme and gives the adversary unrestricted access to the encryption/decryption oracles. Second,

(n − λ)CAKE relaxes the notion of all-or-nothing and parameterizes the number of ciphertext blocks that are not given to the adversary. As we will show in Sec- tion 4.2, this relaxation allows us to design encryption modes that are considerably more efficient than existing modes which offer a comparable level of security.

We stress that (n −λ)CAKE does not consider confidentiality against "traditional" adversaries (i.e., adversaries which do not know the encryption key). Indeed, an indadversary is not given the encryption key but has access to all ciphertext blocks. That is, the ind- adversary can compromise all the s storage servers. An (n − λ)CAKE-adversary is given the encryption key

but can access all but λ ciphertext blocks. In practice,

1. Any party with access to all the ciphertext blocks and the encryption key can recover the plaintext.

the (n − λ)CAKE-adversary has the encryption key but can compromise up to s − 1 storage servers. Therefore, we seek an encryption mode with the following properties:

1) must be ind secure against an adversary which does not know the encryption key but has access to all ciphertext blocks (cf. Definition 1), by compromising all storage servers.

2) must be (n − λ)CAKE secure against an adversary which knows the encryption key but has access to n − λ ciphertext blocks (cf. Definition3), since it cannot compromise all storage servers.

REMARK 4. Property 2 ensures data confidentiality against the attacker model outlined in Section 3.2. Nevertheless, we must also account for weaker ad- versaries (i.e., traditional adversaries) that do not know the encryption key but can access the entire ciphertext —hence, ind security. Note that if the adversary which has access to the encryption key, can also access all the ciphertext blocks, then no cryptographic mechanism can preserve data confidentiality.

4 BASTION: SECURITY AGAINST KEY EXPO- SURE

In this section, we present our scheme, dubbed Bastion, which ensures that plain text data cannot be recovered as long as the adversary has access to all but two ciphertext blocks—even when the encryption key is exposed. We then analyze the security of Bastion with respect to Definition 1 and Definition3.

Overview

Bastion departs from existing AON encryption schemes. Current schemes require a pre-processing round of block cipher encryption for the AONT, followed by another round of block cipher encryption (cf. Figure 2 (a)). Differently, Bastion first encrypts the data with one round of block cipher encryption, and then applies an efficient linear post-processing to the ciphertext (cf. Figure 2 (b)). By doing so, Bastion relaxes the notion of all-or-nothing encryption at the benefit of increased performance (see Figure2).

More specifically, the first round of Bastion consists of CTR mode encryption with a randomly chosen key K, i.e., y ´ = Enc(K, x). The output ciphertext y ´ is

then fed to a linear transform which is inspired by the scheme of [28]. Namely, our transform basically computes y = y ´ • A where A is a square matrix such that:(i) all diagonal elements are set to 0, and (ii) the remaining off-diagonal elements are set to 1. As we shown later, such a matrix is invertible and has the nice property

that A−1 = A. Moreover, y = y ´ • A ensures that each input block y ´ j will depend on all output blocks yi except from yj. This transformation—combined with the fact that the original input blocks have high entropy (due to semantic secure encryption)—result in an indsecure and (n − 2)CAKE secure encryption mode. In the following section, we show how to efficiently compute y ´ • A by means of bitwise XOR operations.

Bastion: Protocol Specification

We now detail the specification of Bastion.

On input a security parameter k, the key generation algorithm of Bastion outputs a key K ∈ {0, 1}k for the underlying block-cipher. Bastion leverages block cipher encryption in the CTR mode, which on input a plaintext bitstream x, divides it in blocks x[1],...,x[m], where m is odd2 such that each block has size l.3 The set of input blocks is encrypted under key K, resulting

in ciphertext y ´ = y ´ [1], . . . , y ´ [m + 1], where y ´ [m + 1] is an initialization vector which is randomly chosen from

{0, 1}l.

Next, Bastion applies a linear transform to y ´ as follows. Let n = m + 1 and assume A to be an n- by- n matrix where element ai,j = 0lif i = j or ai,j = 1l, otherwise.4 Bastion computes y = y ´ • A, where additions and multiplications are implemented by means of XOR and AND operations, respj=enct ´ ively.

That is, y[i] ∈y is computed as y[i] = j=1 (y [j]∧aj,i),

for i = 1 . . . , n.

Given key K, inverting Bastion entails computing y ´ = y • A−1 and decrypting y ´ using K. Notice that matrix A is invertible and A = A−1. The pseudo code of the encryption and decryption algorithms of Bastion are shown in Algorithms 1 and 2, respectively. Both algorithms use F to denote a generic block cipher (e.g., AES).

In our implementation, we efficiently compute the linear transform using 2n XOR operations as follows:

t=y ´ [1]⊕y ´ [2]⊕ • • • ⊕y ´ [n], y[i] = t ⊕y ´ [i], 1 ≤ i ≤n.

Note that y ´ [1] . . . y ´ [n] (computed up to line 6 in Algo- rithm 1) are the outputs of the CTR encryption mode, where y ´ [n] is the initialization vector. Similar to the

CTR encryption mode, the final output of Bastion is one block larger than the original input.

Correctness Analysis

We show that for every x ∈ {0, 1}lm where m is odd, and for every K ∈ {0, 1}l,we have x = Dec(K,Enc(K,x)).

In particular, notice that lines 2-6 of Algorithm 1 and lines 9-12 of Algorithm 2 correspond to the standard CTR encryption and decryption routines, respectively.

1. This requirement is essential for the correctness of the sub- sequent linear transform on the ciphertext blocks. That is, if m is even, then the transform is not invertible.
2. l is the block size of the particular block cipher used.
3. 0l and 1l denote a bitstring of l zeros and a bitstream of ones, respectively.

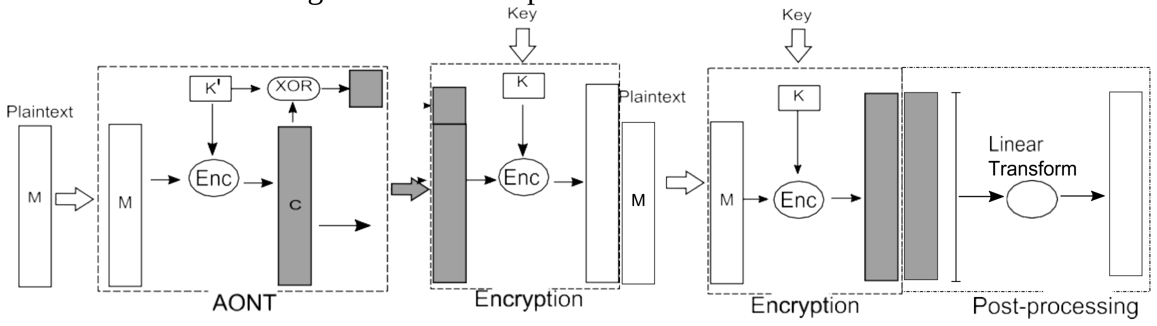


Fig.2.(a)CurrentAONencryptionschemesrequireapre-processingroundofblockcipherencryptionfortheAONT,followedbyanother round of block cipher encryption. (b) On the other hand, Bastion first encrypts the data with one round of block cipher encryption, and then applies an efficient linear post-processing to theciphertext

Algorithm 1 Encryption in Bastion.

```
1: procedure Enc(K,x =x[1]...x[m])
2:     n = m +1
3:     y[n] ←{0,1}^l                    ◁y'[n]istheIVforCTR
4:     fori=1...n−1do
5:         y′[i] = x[i] ⊕ F_K(y′[n]+i)
6:     endfor
7:     t =0^l
8:     fori=1...ndo
9:         t = t⊕y′[i]
10:    endfor
11:    fori=1...ndo
12:        y[i]=y′[i]⊕t
13:    endfor
14:    returny                          ◁ y = y[1] . . .y[n]
15: end procedure
```

Algorithm 2 Decryption in Bastion.

```
1:procedureDec(K,y=y[1]...y[n]) 2: t =0^l
3: for i = 1 . . . n do
4:         t = t ⊕y[i]
5:     endfor
6:  for i = 1 . . . n do 7:
           y′[i]  =  y[i] ⊕t 8: endfor
9: for i = 1 . . . n − 1 do
10:        x[i]=y′[i]⊕F_K^{-1}(y′[n]+i)
11: end for
12:    returnx                          ◁x=x[1]...x[n −1]
13: end procedure
```

Therefore, we are only left to show that the linear transformation computed in lines 7-14 of Algorithm 1 is correctly reverted in lines 2-8 of Algorithm 2. In other words, we need to show that t = i=1..n y[i] (as computed in the decryption algorithm) matches t = i=1..n y [i] ´ (as computed in the encryption algo- rithm).

Recall that t can be computed as follows:

$$t = \bigoplus_{i=1..n} y[i]$$

$$= \bigoplus_{i=1..n} (y'[i] \oplus t)$$

$$= \bigoplus_{i=1..n} y'[i] \oplus \bigoplus_{i=1..n} y'[i]$$

$$= \bigoplus_{i=1..n} \bigoplus_{j=1..n, j \neq i} y[j]$$

$$= \bigoplus_{i=1..n} y'[i]$$

Notice that the last step holds because n is even and therefore each y ´ [j] is XORed for anodd number oftimes.

REMARK 5. We point out that Bastion is not restricted to the CTR encryption mode and can be instanti- ated with other ind-secure block cipher (and stream ciphers) modes of encryption (e.g., CBC,OFB).

Tointerface with our cloud storage model described in Section 3.1, we assume that each user encrypts the data using Bastion before invoking the write() routine. Morespecifically,letEnc(K,•),Dec(K,•)denotethe encryption and decryption routines of Bastion,respec- tively. Given encryption key K and a file f, the user computes v ← Enc(K,f) and invokes write(v) in order touploadtheencryptedfiletothecloud.Inthissettin g, key K remains stored at the user ’ smachine.Similarly, todownloadthefilefromthecloud,theuserinvokes read(•)tofetchvandrunsf←Dec(K,v)torecoverf.

Security Analysis

In this section, we show that Bastion is mathrmind secure and (n − 2)CAKE secure.

LEMMA 1. Bastion is ind secure.

Proof 1. Bastion uses an ind secure encryption mode to encrypt a message, and then applies a linear transform on the ciphertext blocks. It is straight- forward to conclude that Bastion is ind secure. In other words, a polynomial-time algorithm A that has non-negligible advantage in breaking the ind security of Bastion can be used as a black-boxby another polynomial-time algorithm B to breakthe ind security of the underlying encryption mode. In particular, B forwards A's queries to its oracle and applies the linear transformation of Algorithm 1 lines 7-14 to the received ciphertext before forward- ing it to A. The same strategy is used when A outputs two messages at the end of the find stage: the two messages are forwarded to B' s oracle; upon receiving the challenge ciphertext, B applies the linear transformation and forwards it to A. When A replies with its guess b ´ , B outputs the same guess. It is easy to see that if A has non-negligible advantage in guessing correctly which messagewas encrypted, so does B. Furthermore, the runningtime LEMMA 3. Bastion is (n − 2)CAKE secure.

Proof 3. The security proof of Bastion resembles the standard security proof of the CTR encryption mode and relies on the existence of pseudo-random permutations.Inparticular,givenapolynomial-type algorithm A which has non-negligible advantage in the (n − λ)CAKE experiment with λ = 2, we can construct a polynomial-time algorithm B which has non-negligible advantage in distinguishing between a true random permutation and a pseudo-random permutation.

B has access to oracle O and uses it to answer the encryption and decryption queries issued by A. In particular, A's queries are answered as follows:

- Decryption query for y[1]. . . y[n]

1) Computet =y[1]⊕...⊕y[n]
2) Compute y ´ [i] = y[i] ⊕t, for 1 ≤ i≤ n
3) Compute x[i] = y ´ [i] ⊕O(y ´ [n] + i), for 1≤ i ≤ n −1
4) Returnx[1]...x[n−1]

- Encryptionqueryforx[1]...x[n−1]

1) Pick random y ´ [n] ∈ {0,1}l
2) Compute y ´ [i] = x[i] ⊕O(y ´ [n] + i), for 1≤ i ≤ n −1
3) Computet=y ´ [1]⊕...⊕y ´ [n]
4) Compute y[i] = y ´ [i] ⊕t, for 1 ≤ i≤ n
5) Returny[1]...y[n]

When A outputs two messages x1[1] . . . x1[n−1] and x2[1] . . . x2[n − 1], B picks b ∈ {0, 1} at random and does the following:

1) Pick random y ´ [n] ∈ {0,1}l

2) Computey[i]=xb[i]⊕O(y[n],i),for1≤i≤n−1

3) Compute t = yb[1] ⊕. . . ⊕yb[n]

4) Compute yb[i] = y ´ b[i] ⊕t, for 1 ≤ i ≤n

At this point, A selects (n − 2) indexes i1, . . . in−2 and B returns the corresponding yb[i1], . . . , yb[in−2]. Encryption and decryption queries are answered as above. When A outputs its answer b ´ , B outputs 1

if b = b ´ , and 0 otherwise. It is straightforward to see that if A has advantage larger than negligible to guess b, then B has advantage larger than negligible to distinguish a true random permutation from a pseudorandom one. Furthermore, the number of queries issued by B to its oracle amounts to the number of encryption and decryption queriesissued by A. Note that by Lemma 2, during the guess stage, A cannot issue a decryption query on the challenge ciphertext since with only (n − 2) blocks, finding the remaining blocks is infeasible.

REMARK 6. Bastion is not (n − 1)CAKE secure. As shown in the proof of Lemma 2, the adversary can recover one block of y ´ given any (n − 1) blocks of y. If

the adversary recovers y ´ [n] that is used as an IV in the CTR encryption mode, the adversary can easily win the (n − 1)CAKE game. Recall that our security definition allows the adversary to learn the encryptionkey.

REMARK 7. Bastion is (n − 2)CAKE secure according to Definition 3. However, in a practical deployment, we expect that each file spans several thousands blocks 5.When those blocks are evenly spread across servers, each server will store a larger number of

blocks. Therefore, an (n − 2)CAKE secure scheme such as Bastion clearly preserves data confidential- ity unless all servers are compromised.

4. For example, a 10MB file encrypted using AES has morethan 600Kblocks.

TABLE 1

ComparisonbetweenBastionandexistingconstructs.Weassumeaplaintext ofm=n−1blocks.Sinceallschemesaresymmetric, weonlyshow thecomputationoverheadfortheencryption/encodingroutineinthecolumn“Computation”(“b.c.”isthenumberofblock cipher operations; “XOR” is the number of XORoperation

⋆Recall that an ind-adversary can access all storage servers to fetch all ciphertext blocks. Therefore, the adversary can also fetch all the key shares and compute the encryption key.

IV.COMPARISON TO EXISTING SCHEMES

In what follows, we briefly overview several encryption modes and argue about their security (according to Definitions 1 and 3) and performance when compared to Bastion.

CPA-encryption modes

Traditional CPA-encryption modes, such as the CTR mode, provide ind security but are only 1CAKE secure. That is, an adversary equipped with the encryption key must only fetch two ciphertext blocks to break data confidentiality.6

CPA-encryption and secret-sharing

Another option is to rely on the combination of CPA secure encryption modes and secret-sharing.

If the file f is encrypted and then shared with ann-out-of-nsecret-sharingscheme(denotedas“encrypt- then-secret-share” in the following), then the construc- tion is clearly (n − 1)CAKE secure and is also ind secure. However, secret-sharing the ciphertext comes at considerable storage costs; for example, each share would be as large as the file f using a perfect secret sharing scheme—which makes it impractical for storing large files.

Secret-sharing the encryption key and dispersing its shares across the storage servers alongside the cipher-textisnotsecureagainstanind-adversary.Indeed,if the adversary can access all the storage servers and down- load all ciphertext blocks, the adversary may as well download all key shares and compute the encryption key.

1. WeassumethattheCTRencryptionroutine startswitha random IV that is incremented at

every blockencryption.

AON encryption

Recall that an AONT is not an encryption scheme and does not require the decryptor to have any secret key. That is, an AONT is not secure against an ind-adversary which can access all the ciphertext blocks. One alter- native is to combine the use of AONT with standard encryption. Rivest [26] suggests to pre-process a mes- sage with an AONT and then encrypt its output with an encryption mode. This paradigm is referred to in the literature as AON encryption and provides(n−1)CAKE security. Existing AON encryption schemes requireat least two rounds of block cipher encryption with two different keys [12], [26]. At least one round is required for the actual AONT that embeds the first encryption key in the pseudo-ciphertext (cf. Section 2). An addi- tional round uses another encryption key that is kept secret to guarantee CPA-security. However, two encryp- tion rounds constitute a considerable overhead when encrypting and decrypting large files. In Appendix A, we describe possible ways of modifying the AONTs of [26] and [12] to achieve ind security and (n −1)CAKE

security without adding another round of blockcipher

encryption, and we discuss their shortcomings.

Clearly, these solutions are either not satisfactory in terms of security or incur a large overhead when compared to Bastion and may not be suitable to store large files in a multi-cloud storagesystem.

Performance Comparison

Table 1 compares the performance of Bastion with the encryption schemes considered so far, in terms of computation, storage, andsecurity.

Given a plaintext of m blocks, the CTR encryption mode outputs n = m + 1 ciphertext blocks, computed with (n − 1) block cipher operations and (n − 1) XOR

operations. The CTR encryption mode is *ind* secure but only 1*CAKE* secure.

Rivest AONT outputs a pseudo-ciphertext of $n = m + 1$ blocks using $2(n − 1)$ block cipher operations and $3(n−1)$ XOR operations. Desai AONT outputs the same number of blocks but requires only $(n − 1)$ block cipher operations and $2(n − 1)$ XOR operations. Both Rivest AONT and Desai AONT are, however, not *ind* secure since the encryption key used to compute the AONT output is embedded in the output itself. Encrypting the output of Rivest AONT or Desai AONT with a stan- dard encryption mode (both [12] and [26] use the ECB encryption mode), requires additional n block cipher operations,andyieldsanAONencryptionthatis*ind*

secure7 and $(n − 1)$*CAKE* secure. Encrypt-then-secret- share (cf. Section 4.4) is *ind* secure and $(n − 1)$*CAKE* secure. It requires $(n − 1)$ block cipher operations and n XOR operations if additive secret sharing is used. How- ever secret-sharing encryption results in a prohibitively large storage overhead of n^2blocks. Bastion also outputs $n = m + 1$ ciphertext blocks. It achieves *ind* security and $(n − 2)$*CAKE* security with only $(n − 1)$ block cipher operations and $(3n − 1)$ XOR operations.8

We conclude that Bastion achieves a solid tradeoff between the computational overhead of existing AON encryption modes and the exponential storage overhead of secret-sharing techniques, while offering a compa- rable level of security. In Section 6, we confirm the superior performance of Bastion by means of imple- mentation.

V.IMPLEMENTATION AND EVALUATION

In this section, we describe and evaluate a prototype implementation modeling a read-write storage system based on Bastion. We also discuss insights with respect to the integration of Bastion within existing dispersed storage systems.

Implementation Setup

Our prototype, implemented in C++, emulates the read- write storage model of Section 3.1. We instantiate Bastion with the CTR encryption mode (cf. Figure 1) using both AES128 and Rijndael256, implemented using the libmcrypt.so. 4.4.7 library. Since this library doesnot natively support the CTR encryption mode, we use it for the generation of the CTR key stream, which is later XORed with the plaintext.

We compare Bastion with the AON encryption schemes of Rivest [26] and Desai [12]. For baseline comparison, we include in our evaluation the CTR encryption mode and the AONTs due to Rivest [26] and

1. Security according to Definition 1 is achieved because the key used to create the AONT is always random, even if the key used to add the outer layer of encryption isfixed.

2.Bastionrequires(n−1)XORoperationsfortheCTRencrytioand 2n XOR operations for the lineartransform.

Desai [12], which are used in existing dispersed storage systems, e.g., Cleversafe [25]. We do not evaluate the performance of secret-sharing the data because of its prohibitively large storage overhead (squared in the number of input blocks). We evaluate our implemen- tations on an Intel(R) Xeon(R) CPU E5-2470 running at 2.30GHz. Note that the processor clock frequency might have been higher during the evaluation due to the TurboBoost technology of the CPU. In our evaluation, we abstract away the effects of network delays and congestion, and we only assess the processing perfor- mance of the encryption for the considered schemes. This is a reasonable assumption since all schemes are length-preserving (plus an additional block of l bits), and are therefore likely to exhibit the same network performance. Moreover, we only measure the per- formance incurred during encryption/encoding, since all schemes are symmetric, and therefore the decryp- tion/decoding performance is comparable to that of the encryption/encodingprocess.

We measure the peak through put and the latency ex- hibited by our implementations w.r.t. various file/block sizes. For each data point, we report the average of 30 runs. Due to their small widths, we do not show the corresponding 95% confidence intervals.

Evaluation Results

Our evaluation results are reported in Figure 3 and Figure 4. Both figures show that Bastion considerably improves (by more than 50%) the performance of ex- isting (n − 1)CAKE encryption schemes and only in- curs a negligible overhead when compared to existing semantically secure encryption modes (e.g., the CTR encryption mode) that are only 1CAKE secure.

In Figure 3, we show the peak throughput achie- ved by the CTR encryption mode, Bastion, Desai AONT/AON, and Rivest AONT/AON schemes. The peak throughput achieved by Bastion reaches almost 72 MB/s and is only 1% lower than the one exhibited by the CTR encryption mode. When compared with ex- isting (n − 1)CAKE secure schemes, such as DesaiAON encryption and Rivest AON encryption, our results show that the peak throughput of Bastion is almost twice as large as that of Desai AON encryption, and more than three times larger than the peak throughput of Rivest AON encryption.

We also evaluate the performance of Bastion, with respect to different block sizes of the underlying block cipher.Our results show that—irrespective of the block size—Bastion only incurs a negligible performance de- terioration in peak throughput when compared to the CTR encryption mode. Figures 4(a) and 4(b) show the latency (in ms) incurred by the encryption/encoding routines for different file sizes. The latency of Bastion is comparable to that of the CTR encryption mode—for both AES128 and Rijandael256—and results in a con- siderable improvement over existing AON encryption schemes (more than 50% gain inlatency).

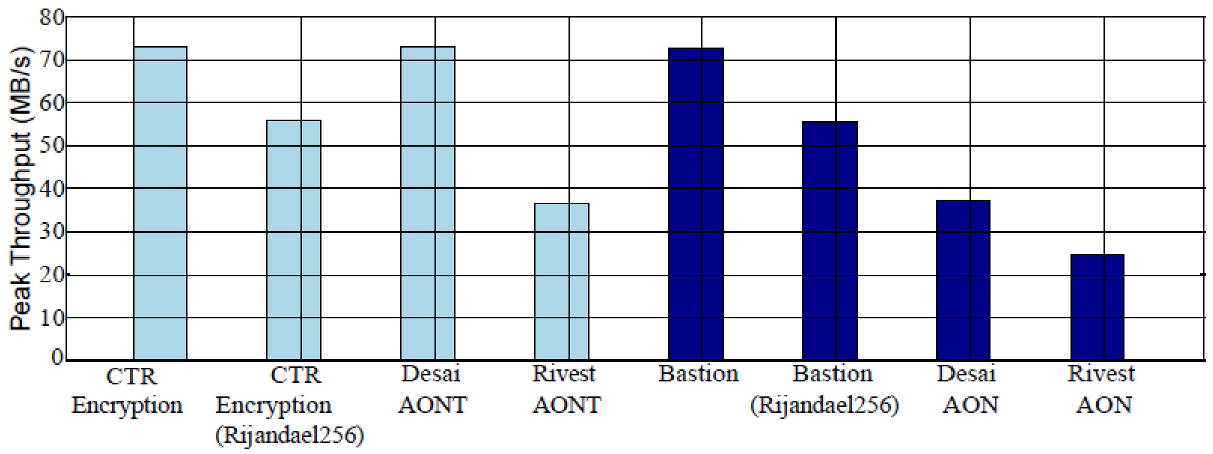
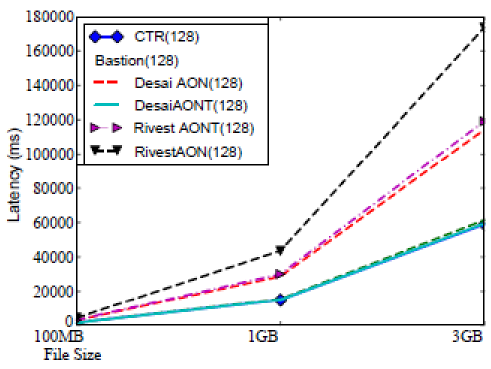
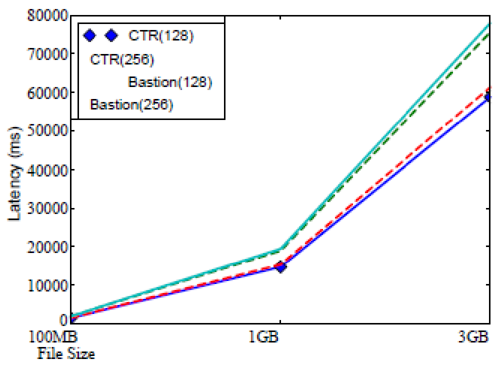


Fig. 3. Peak throughput comparison. Unless otherwise specified, the underlying block cipher is AES128. Each data point is averaged over30runs.Histogramsindarkbluedepict encryptionmodeswhichoffercomparablesecuritytoBastion.Lightbluehistogramsreferto encryption/encoding modes where individual ciphertext blocks can be inverted when the key is exposed.



(a) Latency of encryption/encoding for differ- ent filesizes.



(b) Latency of encryption/encoding fordif- ferent block sizes of the underlying block cipher.

Fig. 4. Performance evaluation of Bastion. Each data point in is averagedover30runs.Unlessotherwisespecified,t heunderlying blockcipherisAES-128.CTR(256)and Bastion(256)denote the

CTR encryption mode and Bastion encryption routine,respectively, instantiated with Rijandael256.

Deployment within HYDRAstor

Recall that Bastion preserves data confidentiality against an adversary that has the encryption key as long as the adversary does not have access to two ciphertext blocks. In a multi-cloud storage system, if each server stores at least two ciphertext blocks, then Bastion clearly preserves data confidentiality unless all servers are compromised.

In scenarios where servers can be faulty, Bastion can be combined with information dispersal algorithms (e.g., [24]) to provide data confidentiality and fault tolerance. Recall that information dispersal algorithms (IDA), parameterized with t1, t2 (where t1 ≤ t2), encode data into t2 symbols such that the original data can be recovered from any t1 encoded symbols. In our multi- cloud storage system (cf. Section 3.1), the ciphertext

output by Bastion is then fed to the IDA encoding routine, with symbols of size l bits, and with parameters

t2 ≥ 2s, t1 < t2, where s is the number of available servers. Since the output of the IDA is equally spread across the s servers, by setting t2 ≥ 2s, we ensure

that each server stores at least two ciphertext blocks worth of data. Finally, the encoded symbols are input to the write() routine that distributes symbols evenly to each of the storage servers. Recovering f via the read() routine entails fetching t1 encoded symbols from the servers and decoding them via the IDA decoding routine. The resulting ciphertext can be decrypted using Bastion to recover file f . By doing so, data confiden- tiality is preserved even if the key is exposed unless

t= st1t2 servers are compromised. Furthermore, data availability is guaranteed in spite of (s − t) server failures.

HYDRAstor

We now discuss the integration of a prototype im- plementation of Bastion within the HYDRAstor grid storage system [13], [23]. HYDRAstor is a commercial secondary storage solution for enterprises, which consists of a back-end architectured as a grid of stor- age nodes built around a distributed hash table. HY- DRAstor tolerates multiple disk, node and network failures, rebuilds the data automatically after failures, and informs users about recoverability of the deposited data [13]. The reliability and availability of the stored data can be dynamically adjusted by the clients with each write operation, as the back-end supports multiple data resiliency classes[13].

HYDRAstor distributes written data to multiple disks using the distributed resilient data technology (DRD); the combination of Bastion with DRD ensures that an adversary which has the encryption key and compromises a subset of the disks (i.e., determined by the reconstruction threshold), cannot acquire any meaningful information about the data stored on the disk. To better assess the performance impact of Bastion in HYDRAstor, we evaluated the performance of Bastion in the newest generation HYDRA stor HS8-4000 series system, which uses CPUs with accelerated AES encryption (i.e., the AESNI instruction set). In our experiments, all written data was unique to remove the effect of data de duplication. Results show that the write bandwidth was not affected by the integration of Bastion. The read bandwidth decreased only by 3%. In both read and write operations, the CPU utilization in the system only increased marginally. These experiments clearly suggest that Bastion can be integrated in existing commercial storage systems to strengthen the security of these systems under key exposure, without affecting performance.

VI. RELATEDWORK

To the best of our knowledge, this is the first work that addresses the problem of securing data stored in multi- cloud storage systems when the cryptographic material is exposed. In the following, we survey relevant related work in the areas of deniable encryption, information dispersal, all-or-nothing transformations, secret-sharing techniques, and leakage-resilient cryptography.

Deniable Encryption

Our work shares similarities with the notion of “shared- key deniable encryption” [9], [14], [18]. An encryption scheme is “deniable” if—when coerced to reveal the encryption key—the legitimate owner reveals “fake keys” thus forcing the ciphertext to “look like” the encryption of a plaintext different from the original one—hence keeping the original plaintext private. Deniable en- cryption therefore aims to deceive an adversary which does not know the “original” encryption key but, e.g., can only acquire “fake” keys. Our security definition models an adversary that has access to the real keying material.

Information Dispersal

Information dispersal based on erasure codes [30] has been proven as an effective tool to provide reliability in a number of cloud-based storage systems [1], [2], [20], [33]. Erasure codes enable users to distribute their data on a number of servers and recover it despite some servers failures.

Ramp schemes [7] constitute a trade-off between the security guarantees of secret sharing and the efficiency of information dispersal algorithms. A ramp scheme achieves higher “code rates” than secret sharing and features two thresholds $t1$, $t2$. At least $t2$ shares are required to reconstruct the secret and less than $t1$ shares provide no information about the secret; a number of shares between $t1$ and $t2$ leak “some” information.

All or Nothing Transformations

All-or-nothing transformations (AONTs) were first introduced in [26] and later studied in [8], [12]. The majority of AONTs leverage a secret key that is em- bedded in the output blocks. Once all output blocks are available, the key can be recovered and single blocks can be inverted. AONT, therefore, is not an encryption scheme and does not require the decryptor to have any key material. Resch et al. [25] combine AONT and information dispersal to provide both fault-tolerance and data secrecy, in the context of distributed storage systems. In [25], however, an adversary which knows the encryption key can decrypt data stored on single servers.

Secret Sharing

Secret sharing schemes[5]allow a dealer to distribute a secret among a number of shareholders, such that only authorized subsets of shareholders can reconstruct the secret .In threshold secret sharing schemes[11],[27],the dealer defines a threshold *t* and each set of shareholders of cardinality equal to or greater than *t* is authorized to reconstruct the secret. Secret sharing guarantees security against a non-authorized subset of shareholders; however, they incur a high computation/storage cost, which makes them impractical for sharing large files. Rabin [24] proposed an information dispersal algorithm with smaller overhead than the one of [27], however the proposal in [24] does not provide any security guarantees when a small number of shares (less than the reconstruction threshold) are available.Krawczyk

[19] proposed to combine both Shamir's [27] and Ra- bin's [24] approaches; in [19] a file is first encrypted using AES and then dispersed using the scheme in [24], while the encryption key is shared using the scheme in [27]. In Krawczyk's scheme, individual ciphertext blocks encrypted with AES can be decrypted once the key is exposed.

Leakage-resilient Cryptography

Leakage-resilient cryptography aims at designing cryptographic primitives that can resist an adversary which learns partial information about the secret state of a sys- tem, e.g., through side-channels [22]. Different models allow to reason about the "leaks" of real implementations of cryptographic primitives [22]. All of these models, however, limit in some way the knowledge of the secret state of a system by the adversary. In contrast, the adversary is given all the secret material in our model.

VII. CONCLUSION

In this paper, we addressed the problem of securing data outsourced to the cloud against an adversary which has access to the encryption key. For that purpose, we introduced a novel security definition that captures data confidentiality against the new adversary .We then proposed Bastion, a scheme which ensures the confidentiality of encrypted data even when the adversary has the encryption key, and all but *two* cipher-text blocks. Bastion is most suitable for settings where the ciphertext blocks are stored in multi-cloud storage systems. In these settings, the adversary would need to acquire the encryption key, and to compromise *all* servers, in order to recover any single block of plaintext.

We analyzed the security of Bastion and evaluated its performance in realistic settings. Bastion consider- ably improves (by more than 50%) the performance of existing primitives which offer comparable security under key exposure, and only incurs a negligible overhead (less than 5%) when compared to existing semantically secure encryption modes (e.g., the CTR encryption mode). Finally, we showed how Bastion can be practically integrated within existing dispersed storage systems.

REFERENCES

[1] M. Abd-El-Malek, G. R. Ganger, G. R. Goodson, M. K. Reiter, and J. J. Wylie, "Fault-Scalable Byzantine Fault-Tolerant Services," in ACM Symposium on Operating Systems Principles (SOSP), 2005, pp.59–74.

[2] M. K. Aguilera, R. Janakiraman, and L. Xu, "Using Erasure Codes Efficiently for Storage in a Distributed System," in International Conference on Dependable Systems and Networks (DSN), 2005, pp.336–345.

[3] W. Aiello, M. Bellare, G. D. Crescenzo, and R. Venkatesan, "Security amplification by composition: The case of doubly- iterated, ideal ciphers," in Advances in Cryptology (CRYPTO), 1998, pp.390–407.

[4] C. Basescu, C. Cachin, I. Eyal, R. Haas, and M. Vukolic, "Ro- bust Data Sharing with

Key-value Stores," in ACM SIGACT- SIGOPS Symposium on Principles of Distributed Computing (PODC), 2011, pp.221–222.

[5] A. Beimel, "Secret-sharing schemes: A survey," in Interna- tional Workshop on Coding and Cryptology (IWCC), 2011, pp. 11–46.

[6] A. Bessani, M. Correia, B. Quaresma, F. André, and P. Sousa, "DepSky: Dependable and Secure Storage in a Cloud-of- clouds," in Sixth Conference on Computer Systems (EuroSys), 2011, pp.31–46.

[7] G. R. Blakley and C. Meadows, "Security of ramp schemes," in Advances in Cryptology (CRYPTO), 1984, pp.242–268.

[8] V. Boyko, "On the Security Properties of OAEP as an All- or- nothing Transform," in Advances in Cryptology (CRYPTO), 1999, pp.503–518.

[9] R. Canetti, C. Dwork, M. Naor, and R. Ostrovsky, "Deniable Encryption," in Proceedings of CRYPTO,1997.

[10]Cavalry, "Encryption Engine Dongle," http://www. cavalrystorage.com/en2010.aspx/.

[11]C. Charnes, J. Pieprzyk, and R. Safavi-Naini, "Conditionally secure secret sharing schemes with disenrollment capability," in ACM Conference on Computer and Communications Security (CCS), 1994, pp.89–95.

[12]A. Desai, "The security of all-or-nothing encryption: Protecting against exhaustive key search," in Advances in Cryptology (CRYPTO), 2000, pp.359–375.

[13]C. Dubnicki, L. Gryz, L. Heldt, M. Kaczmarczyk, W. Kil- ian, P. Strzelczak, J. Szczepkowski, C. Ungureanu,and

M. Welnicki, "HYDRAstor: a Scalable Secondary Storage," in USENIX Conference on File and Storage Technologies (FAST), 2009, pp.197–210.

[14]M. Dürmuth and D. M. Freeman, "Deniable encryption with negligible etection probability: An interactive construction," in EUROCRYPT, 2011, pp.610–626.

[15]EMC, "Transform to a Hybrid Cloud," http://www.emc. com/campaign/global/hybridcloud/index.htm.

[16]IBM, "IBM Hybrid Cloud Solution," http://www-01.ibm. com/software/tivoli/products/hybrid-cloud/.

[17]J. Kilian and P. Rogaway, "How to protect DES against exhaustive key search," in Advances in Cryptology (CRYPTO), 1996, pp.252–267.

[18]M. Klonowski, P. Kubiak, and M. Kutylowski, "Practical De- niable Encryption," in Theory and Practice of Computer Science (SOFSEM), 2008, pp.599–609.

[19]H. Krawczyk, "Secret Sharing Made Short," in Advances in Cryptology (CRYPTO), 1993, pp.136–146.

[20]J. Kubiatowicz, D. Bindel, Y. Chen, S. E. Czerwinski, P. R. Eaton,D.Geels, R.Gummadi, S.C.Rhea, H.Weatherspoon,

W. Weimer, C. Wells, and B. Y. Zhao, "Ocean Store: An Architecture for Global-Scale Persistent Storage," in International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS), 2000, pp. 190–201.

[21]L. Lamport, "On interprocess communication,"1985.

[22]S. Micali and L. Reyzin, "Physically observable cryptography (extended abstract)," in Theory of Cryptography Conference (TCC), 2004, pp.278–296.

[23]NEC Corp., "HYDRAstor Grid Storage," http://www.hydrastor.com.

[24]M. O. Rabin, "Efficient dispersal of information for security, load balancing, and fault tolerance," J. ACM, vol. 36, no. 2, pp. 335–348,1989.

[25]J. K. Resch and J. S. Plank, "AONT-RS: Blending Security and Performance in Dispersed Storage Systems," in USENIX Conference on File and Storage Technologies (FAST), 2011, pp. 191–202.

[26]R. L. Rivest, "All-or-Nothing Encryption and the Package Transform," in International Workshop on Fast Software Encryption (FSE), 1997, pp.210–218.

[27]A. Shamir, “How to Share a Secret?” in Communications of the ACM, 1979, pp.612–613.

[28]D. R. Stinson, “Something About All or Nothing (Trans- forms),” in Designs, Codes and Cryptography, 2001, pp. 133– 138.

[29]StorSimple, “Cloud Storage,”http://www.storsimple.com/.

[30]J. H. van Lint, Introduction to Coding Theory. Secaucus, NJ, USA: Springer-Verlag New York, Inc.,1982.

[31]Wikipedia, “Edward Snowden,” http://en.wikipedia.org/ wiki/Edward_Snowden#Disclosure.

[32]Z. Wu, M. Butkiewicz, D. Perkins, E. Katz-Bassett, and H. V.Madhyastha,“SPANStore:Cost-effectiveGeo-replicated Storage Spanning Multiple Cloud Services,” in ACM Symposium on Operating Systems Principles (SOSP), 2013, pp.292–308.

[33]H. Xia and A. A. Chien, “RobuSTore: a Distributed Stor- age Architecture with Robust and High Performance,” in ACM/IEEE Conference on High Performance Networking and Computing (SC), 2007, p.