Securing Data Under Key Exposure

Evangelin Sonia.S.V[1], Infantine Ruth Monita.M[25]
[1,] Assistant Professor, Dept of Computer Science And Engineering
[2]Dept of Computer Science And Engineering
[1, 2] Sri Shakthi Institute Of Engineering And Technology,Coimbatore

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 ciphertext. This may be achieved, for example, by spreading ciphertext blocks across servers in multiple administrative domains—thus assuming that the adversary cannot compromise all 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 ciphertext 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 ciphertext 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 ciphertext 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 BASTION in commercial dispersed storage systems. Our evaluation results suggest that BASTION is well-suited for integration in existing systems since it incurs less than 5% overhead compared to existing semantically secure encryption modes.*

Keywords- Key exposure, data confidentiality, dispersed storage.

I. INTRODUCTION

The 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 ciphertext, 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 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 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 invalidates the security of most 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 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 ciphertext 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 HYDRAstor grid storage system [13], [23].

The remainder of the paper is organized as follows. In Section 2, 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 P_l is the space of all $(2^l)!$ l-bits permutations, then for any $a \in \{0, 1\}^k$, we have
F $(a, \cdot) \in P_l$. We also write $F_a(x)$ 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 kand l. For a given block cipher $F \in$ BC(k, l), we denote $F^{-1} \in$ BC(k, l) as $F^{-1}(a, y)$ or as $F_a^{-1}(y)$, for
a $\in \{0, 1\}^k$.

2.1 Encryption modes

An encryption mode based on a block cipher F/F^{-1} is given by a triplet of algorithms

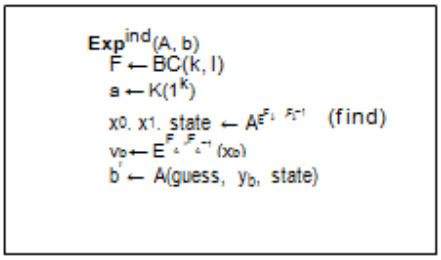
$\varrho = (K, E, D)$ where:
K The key generation algorithm is a probabilistic algorithm which takes as input a security parameter k and outputs a key
a $\in \{0, 1\}^k$ that specifies F_a and F_a^{-1}.

E The encryption algorithm is a probabilistic algorithm which takes as input a message

$x \in \{0, 1\}^*$, and uses F_a and F_a^{-1} as oracles to output ciphertext y.

D The decryption algorithm is a deterministic algorithm which takes as input a ciphertext y, and uses F_a and F_a^{-1} as oracles to output plaintext $x \in \{0, 1\}^*$, or $\perp$ if y is invalid.

For correctness, we require that for any key $a \leftarrow K(1^k)$, for any message $x \in \{0, 1\}^*$, and for any $y \leftarrow E^{F,F^{-1}}(x)$, we have $x \leftarrow D^{F,F^{-1}}(y)$. A A
Security is defined through the following chosen-plaintext attack (CPA) game adapted for block ciphers:



In the *ind* experiment, the adversary has unrestricted oracle access to $E^{F,F^{-1}}$ during the "find" stage. At this point, A outputs two messages of equal length x_0, x_1, and some state information that are passed as input when the adversary is initialized for the "guess" stage (e.g., state can contain the two messages x_0, x_1). During the "guess" stage, the adversary is given the ciphertext of one message out of x_0, x_1 and must guess which message was actually encrypted. The advantage of the adversary in the *ind* experiment is:

$$Adv^{ind}(A) = |Pr[Exp^{ind}(A, 0) = 1] - Pr[Exp^{ind}(A, 1) = 1]|$$

DEFINITION 1. An encryption mode $\varrho = (K, E, D)$ is *ind* secure if for any probabilistic polynomial time (p.p.t.) adversary A, we have $Adv^{ind}(A) \leq \varrho$, where
ϱ is a negligible function in the security parameter.

REMARK 1. The ind experiment allows the adversary to see the entire (challenge) ciphertext. In a scenario 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 therein.

REMARK 2. In the *ind* experiment (and in other expe iments 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.

2.2 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 AONT is given by a pair of p.p.t. algorithms $\Pi = (E, D)$ where:

E The encoding algorithm is a probabilistic algorithm which takes as input a message $x \in \{0, 1\}^*$, and outputs a pseudo-ciphertext y.

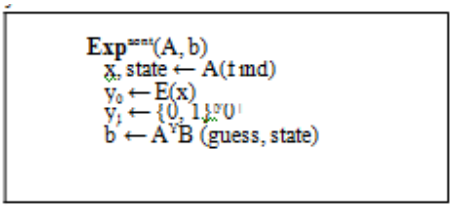
D The decoding algorithm is a deterministic algorithm which takes as input a pseudo-

ciphertext y, and outputs either a message $x \in \{0, 1\}^*$ or $\perp$ to indicate that the input pseudo-ciphertext is invalid.

For correctness, we require that for all $x \in \{0, 1\}^*$, and for all $y \leftarrow E(x)$, we have $x \leftarrow D(y)$.

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 pseudo-ciphertext y can be written as

y = y[1] . . . y[n], where $|y[i]| = l$ and $n \geq 1$.



On input j, the oracle Y_b returns $y_b[j]$ and accepts up to $(n - 1)$ queries. The *aont* experiment models an adversary which must distinguish between the encoding 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:

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

DEFINITION 2. An All-or-Nothing Transform $\Pi = (E, D)$ is *aont* secure if for any p.p.t. adversary A, we have $Adv^{aont}(A) \leq \varrho$, where ϱ is a negligible function in the security parameter.

Known AONTs

Rivest [26] suggested the *package transform* which lever-ages 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 F_K (i), where K is a random key. The n-th output block is computed XORing K with the encryption of each of the previous output blocks, using a key K_0 that is publicly known. That is, given x[1] . . . x[m], the package transform outputs y[1] . . . y[n], with n = m + 1, where:

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

$$y[n] = K \bigoplus_{i=1}^{n-1} F_{K0}(y[i] \oplus i).$$

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.

3.1 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 nowadays [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 $S_1, \ldots, S_s$, 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.

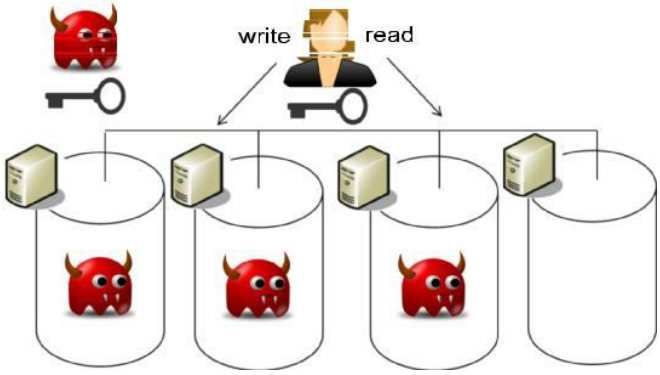


Fig. 1. Our attacker model. We assume an adversary which can acquire all the cryptographic secret material, and can compromise a large fraction (up to all but one) of the storage servers.

3.2 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. Figure 1). 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.

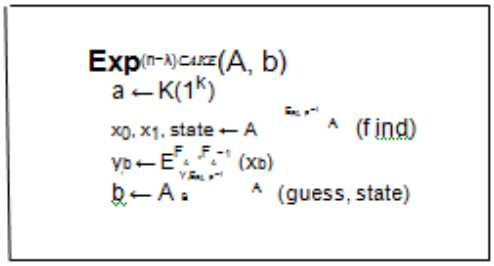
3.3 (n − λ)-CAKE Security

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 achievable[1], 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.

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 ciphertext y can be written as y = y[1] . . . y[n] where |y[i]| = l and n > 1.



The adversary has unrestricted access to $E^F A^{,F} A^{-1}$ in both the "find" and "guess" stages. On input j, the oracle Y_b returns $y_b[j]$ and accepts up to n − λ queries. On the one hand, unrestricted oracle access to $E^F A^{,F} A^{-1}$ captures the adversary's knowledge of the secret key. On the other hand, the oracle Y_b 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:

$$\mathrm{Adv}^{(n-\lambda)CAKE}(A) = \Pr[\mathbf{Exp}^{(n-\lambda)CAKE}(A, 1) = 1] - \Pr[\mathbf{Exp}^{(n-\lambda)CAKE}(A, 0) = 1]$$

DEFINITION 3. An encryption mode $\varrho = (K, E, D)$ is

(n − λ)*CAKE* secure if for any p.p.t. adversary A, we have $\mathrm{Adv}^{(n-\lambda)CAKE}(A) \leq \varrho$, where ϱ 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 Section 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.

1. Any party with access to all the ciphertext blocks and the encryption key can recover the plaintext. the $(n - \lambda)$*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 $(n - \lambda)$*CAKE* secure against an adversary which knows the encryption key but has access to $n - \lambda$ ciphertext blocks

REMARK 4. Property 2 ensures data confidentiality against the attacker model outlined in Section 3.2. Nevertheless, we must also account for weaker 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.

IV. BASTION: SECURITY AGAINST KEY EXPOSURE

In this section, we present our scheme, dubbed Bastion, which ensures that plaintext 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 Definition 3.

4.1 Overview

Bastion departs from existing AON encryption schemes. Current schemes require a pre-processing round of block cipher encryption for the AONT, fol-lowed 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 Figure 2).

More specifically, the first round of Bastion consists of CTR mode encryption with a randomly chosen key K, i.e., $y' = \text{Enc}(K, x)$. The output ciphertext y' is then fed to a linear transform which is inspired by the scheme of [28]. Namely, our transform basically computes $y = y' \cdot A$ 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, $y = y' \cdot A$ ensures that each input block y'_j will depend on all output blocks y_i except from y_j . This transformation—combined with the fact that the original input blocks have high entropy result in an ind-secure and $(n - 2)$*CAKE* secure encryption mode. In the following section, we show how to efficiently compute $y' \cdot A$ by means of bitwise XOR operations.

4.2 Bastion: Protocol Specification

On input a security parameter k, the key generation algorithm of Bastion outputs a key $K \in \{0, 1\}^k$ 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 odd[2] such that each block has size l.[3] The set of input blocks is encrypted under key K, resulting in ciphertext $y' = y'[1], \ldots, y'[m + 1]$, 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 $a_{i,j} = 0^l$ if $i = j$ or $= 1^l$, otherwise.[4] Bastion computes $y = y' \cdot A$,

ai,j

where additions and multiplications are implemented by means of XOR and AND operations, respectively.

That is, $y[i] \in y$ is computed as $y[i] = \bigoplus_{j=1}^{j=n} (y'[j] \wedge a_{j,i})$,

for $i = 1 \ldots, n$.

Given key K, inverting Bastion entails computing $y' = y \cdot A^{-1}$ and decrypting y' using K. Notice that matrix A is invertible and $A = A^{-1}$. The pseudocode 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:

$t = y'[1] \oplus y'[2] \oplus \cdots \oplus y'[n]$, $y[i] = t \oplus y'[i]$, $1 \leq i \leq n$.

4.3 Correctness Analysis

We show that for every $x \in \{0, 1\}^{lm}$ where m is odd, and for every $K \in \{0, 1\}^{l}$

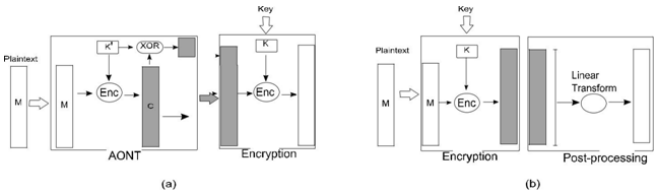


Fig. 2. (a) Current AON encryption schemes require a pre-processing round of block cipher encryption for the AONT, followed by another round of block cipher encryption. (b) On the other hand, BASTION first encrypts the data with one round of block cipher encryption, and then applies an efficient linear post-processing to the ciphertext

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] is the IV for CTR
4:    for i = 1 . . . n − 1 do
5:       y|i| = x|i| ⊕ F_K(y|n| + i)
6:    end for
7:    t = 0^l
8:    for i = 1 . . . n do
9:       t = t ⊕ y|i|
10:   end for
11:   for i = 1 . . . n do
12:      y|i| = y|i| ⊕ t
13:   end for
14:   return y                ◁ y = y|1| . . . y|n|
15:   end procedure
```

Algorithm 2 Decryption in Bastion.

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

Recall that t can be computed as follows:

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

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

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

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

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

Notice that the last step holds because n is even and therefore each $y'[j]$ is XORed for an odd number of times.

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

To interface 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. More specifically, let Enc(K, ·), Dec(K, ·) denote the encryption and decryption routines of Bastion, respectively.

4.4 Security Analysis

In this section, we show that Bastion is mathemind 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-box by another polynomial-time algorithm B to break the *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 message

was encrypted, so does B. Furthermore, the running time of B is the one of A plus the time to apply the linear transformation to A's queries.

LEMMA 2: Given any n − 2 blocks of y[1] . . . y[n] as output by Bastion, it is infeasible to compute any y′[i], for 1 ≤ i ≤ n.

Proof 2: Let y = y[1], . . . , y[n] ← E(K, x=x[1] . . . x[m]). Note that given any (n − 1) blocks of y, the adversary can compute one block of y′.
particular, y′[i] = j=n y[j], for any 1 ≤ i ≤ n.

Definition 3. However, if only (n − 2) blocks of y are given, then each of the n blocks of y′ can take on any possible values in $\{0, 1\}^l$, depending on the two unknown blocks of y. Recall that each block y′[i] is dependent on (n − 1) blocks of y and it is pseudo-random as output by the CTR encryption mode. Therefore, given any (n − 2) blocks of y, then y′[i] could take any of the 2^l possibilities, for 1 ≤ i ≤ n.

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. In particular, given a polynomial-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) Compute t = 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) Return x[1] . . . x[n − 1]

- ***Encryption query for* x[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) Compute t = y′[1] ⊕ . . . ⊕ y′[n]
4) Compute y[i] = y′[i] ⊕ t, for 1 ≤ i ≤ n
5) Return y[1] . . . y[n]

When A outputs two messages $x_1[1] \ldots x_1[n-1]$ and $x_2[1] \ldots x_2[n-1]$, B picks b ∈ {0, 1} at random and does the following:

1) Pick random $y_b'[n] \in \{0, 1\}^l$
2) Compute $y_b'[i] = x_b[i] \oplus O(y_b'[n], i)$, for 1 ≤ i ≤ n−1
3) Compute $t = y_b'[1] \oplus \ldots \oplus y_b'[n]$
4) Compute $y_b[i] = y_b'[i] \oplus t$, for 1 ≤ i ≤ n

At this point, A selects (n − 2) indexes $i_1, \ldots i_{n-2}$ and B returns the corresponding $y_b[i_1], \ldots, y_b[i_{n-2}]$.

TABLE 1 Comparison between BASTION and existing constructs. We assume a plaintext of m = n − 1 blocks. Since all schemes are symmetric, we only show the computation overhead for the encryption/encoding routine in the column "Computation" ("b.c." is the number of block cipher operations; "XOR" is the number of XOR operations.

	Computation	Storage (blocks)	Security
CTR Encryption	n − 1 b.c. n − 1 XOR	n	1*CAKE* ind-secure
Rivest AONT [26]	2(n − 1) b.c. 3(n − 1) XOR	n	N/A ind-INsecure
Desai AONT [12]	n − 1 b.c. 2(n − 1) XOR	n	N/A ind-INsecure
Rivest AON Encryption [26]	3n − 2 b.c. 3(n − 1) XOR	n	(n − 1)*CAKE* ind-secure
Desai AON Encryption [12]	2n − 1 b.c. 2(n − 1) XOR	n	(n − 1)*CAKE* ind-secure
Encrypt-then -secret-share	n − 1 b.c. 2n − 1 XOR	n^2	(n − 1)*CAKE* ind-INsecure*
Bastion	n − 1 b.c. 3n − 1 XOR	n	(n − 2)*CAKE* ind-secure

V. 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 1*CAKE* 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 an n-out-of-n secret-sharing scheme (denoted as "encrypt-then-secret-share" in the following), then the construction 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-text is not secure against an *ind*-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.

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 require at 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 additional round uses another encryption key that is kept secret to guarantee CPA-security. However, two encryption 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 block cipher 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 storage system.

5.1 Performance Comparison

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

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 standard encryption mode (both [12] and [26] use the ECB encryption mode), requires additional n block cipheroperations, and yields an AON encryption that is *ind* secure[7] 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^2 blocks.

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 comparable level of security. In Section 6, we confirm the superior performance of Bastion by means of implementation.

VI. 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.

6.1 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 does not natively support the CTR encryption mode, we use it for the generation of the CTR keystream, 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] andDesai [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

implementations 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 performance 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 performance incurred during encryption/encoding, since all schemes are symmetric, and therefore the decryption/decoding performance is comparable to that of the encryption/encoding process.

We measure the peak throughput and the latency exhibited 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.

6.2 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 existing (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 1*CAKE* secure.

In Figure 3, we show the peak throughput achieved 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 existing (n − 1)*CAKE* secure schemes, such as Desai AON 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 deterioration 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 considerable improvement over existing AON encryption schemes (more than 50% gain in latency).

VII. RELATED WORK

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 encryption 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 t_1, t_2. At least t_2 shares are required to reconstruct the secret and less than t_1 shares provide no information about the secret; a number of shares between t_1 and t_2 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.

VIII. 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. Re-iter, 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," inInternational 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: Protect-ing 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, andM. 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 detection 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, "OceanStore: An Archi-tecture 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 Encryp-tion (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-effective Geo-replicated Stor-age 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," inACM/IEEE Conference on High Performance Networking and Computing (SC), 2007, p. 44.