International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 05 Issue: 08 | Aug 2018 www.irjet.net p-ISSN: 2395-0072

SECURING CLOUD DATA UNDER KEY EXPOSURE

Sandeep kumar D[1], P.R.Rajesh Kumar[2]

[1]*PG Scholar, Dept of Computer science Engineering, Sri Krishna Devaraya University, Anantapuramu, AP, INDIA*
[2]*Asst.Professor, Dept of Computer science Engineering, Sri Krishna Devaraya University, Anantapuramu, AP, INDIA*

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 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.*

Key Words: Key exposure, data confidentiality, dispersed storage.

1.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. 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.

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, 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

2.PRELIMINARIES

We adapt the notation of 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 k and l. For a given block cipher $F \in BC(k,l)$, we denote $F^{-1} \in BC(k,l)$ as $F^{-1}(a,y)$ 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 Q = (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 $y \leftarrow$

$E^{Fa,Fa-1}(x)$, we have $x \leftarrow D^{Fa,Fa-1}(y)$.

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

In the *ind* experiment, the adversary has unrestricted oracle access to $E^{Fa,Fa-1}$ 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:

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

Definition 1. An encryption mode = (K,E,D) is *ind* secure if for any probabilistic polynomial time adversary A, we have

$Adv^{ind}_Q(A) \leq \varrho$, where ϱ is a negligible function in the security parameter.

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 Q = (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 pseudociphertext 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. In this paper, we rely on the definition of 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 $n \geq 1$.

Exp$^{aont}_{Q}(A,b)$ $x,state \leftarrow A(find)$ $y_0 \leftarrow {}^{E}(x)$ $y1 \leftarrow \{0,1\}|y0|$ $b' \leftarrow A^{Yb}(guess,state)$

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:

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

Definition 2. An All-or-Nothing Transform= (E,D)is *aont* secure if for any p.p.t. adversary AQ, we have

$Adv^{aont}_{Q}(A) \leq \varrho$, where ϱ is a negligible function in the security parameter.

Known AONTs

Rivest 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 $F_K(i)$, 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:

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

$$y[n] = K \text{ M } F_{K0}(y[i] \oplus i).$$

3.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 with cloud storage providers such as EMC, IBM, and Microsoft, offering products for multicloud systems

In particular, we consider a system of s storage servers $S_1,...,S_s$, 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 which exports two operations:

write(v)This routine splits v into s pieces $\{v_1,...,v_s\}$ and sends hv_ji to server S_j, for $j \in [1...s]$.

read(·) The read routine fetches the stored value v from the servers. For each $j \in [1...s]$, piece v_j is downloaded from server S_j and all

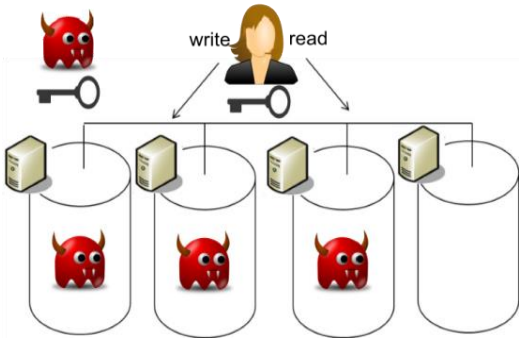

Fig. 1. Our attacker model. We assume an adversary which can acquire all the cryptographic secret material, and can compromise a large fraction (up to all but one) of the storage servers.

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. 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 compromise 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.

access to all ciphertext blocks. That is, the *ind*adversary can compromise all the s storage servers. An $(n - \lambda)$*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 $(n - \lambda)$*CAKE*-adversary has the encryption key but can compromise up to $s - 1$ storage servers. Therefore, properties: we seek an encryption mode with the following does not know the encryption key but has access toQ

1) must be *ind* secure against an adversary which all ciphertext blocks (cf. Definition 1), by compromising all storage servers.

2) must be $(n - \lambda)$*CAKE* secure against an ad-Qversary which knows the encryption key but has access to $n - \lambda$ ciphertext blocks (cf. Definition 2), since it cannot compromise all storage servers.

4.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 2.

4.1 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 . Differently, Bastion first encrypts the data with one round of block cipher encryption, and then applies an efficient linear post-processing to the ciphertext. By doing so, Bastion relaxes the notion of all-or-nothing encryption at the benefit of increased performance.

4.2 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 $K \in \{0,1\}^k$ 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 $y' = y'[1],...,y'[m+1]$, 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 $a_{i,j} = 1^l$, otherwise.[4] Bastion computes $y = y' \cdot A$, 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 $i = 1...,n$.

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: $t = y'[1] \oplus y'[2] \oplus \cdots \oplus y'[n]$,

$$y[i] = t \oplus y'[i],\ 1 \le i \le n.$$

Note that $y'[1]...y'[n]$ (computed up to line 6 in Algorithm 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.

4.3 Correctness Analysis

We show that for every $x \in \{0,1\}^{lm}$ where m is odd, and for every $K \in \{0,1\}^l$, 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 subsequent 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) 0^l and 1^l denote a bitstring of l zeros and a bitstream of l ones, respectively.

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: $y'[n] \leftarrow \{0,1\}^l$ ▯
4: **for** $i = 1...n - 1$ **do** 5: $y'[i] = x[i] \oplus F_K(y'[n] + i)$
end for 6:
$t = 0^l$ 7:
8: **for** $i = 1...n$ **do**
9: $t = t \oplus y'[i]$
10: **end for**
11: **for** $i = 1...n$ **do**
12: $y[i] = y'[i] \oplus t$
13: **end for**
14: **return** y is the IV for CTR ▯ $y = y[1]...y[n]$
15: **end procedure**

Algorithm 2 Decryption in Bastion.

1: **procedure** Dec(K,$y = y[1]...y[n]$)
2: $t = 0^l$
3: **for** $i = 1...n$ **do**
4: $t = t \oplus y[i]$
5: **end for**
6: **for** $i = 1...n$ **do**
7: $y'[i] = y[i] \oplus t$
8: **end for**
9: **for** $i = 1...n - 1$ **do**
10: $x[i] = y'[i] \oplus F_K^{-1}(y'[n] + i)$
11: **end for**
12: **return** x ▯ $x = x[1]...x[n - 1]$
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 (as computed in the decryption algorithm) matchesL other words, we need to show that $t = {}_{i=1..n}\, y[i]$ $t = {}_{i=1..n}\, y'[i]$ (as computed in the encryption algo-

4.4 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 straightforward 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 forwarding it to A. The same strategy is used when A outputs two messages at the end of the *find* stage.

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 $1 \le i \le n$.

Proof 2. Let $y = y[1],...,y[n] \leftarrow E(K,x = x[1]...x[m])$. Note that given any $(n - 1)$ blocks of y, the adversary can compute one block of y'. In particular, $y'[i] = \bigoplus_{j=1,j\neq i}^{j=n} y[j]$, for any $1 \le i \le n$.

As it will become clear later, with one block $y'[i]$ and the encryption key, the adversary has non-negligible probability of winning the game of 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 $1 \le i \le n$.

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 $(n - \lambda)$*CAKE* experiment with $\lambda = 2$, 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 $t = y[1] \oplus ... \oplus y[n]$
  2) Compute $y'[i] = y[i] \oplus t$, for $1 \le i \le n$
  3) Compute $x[i] = y'[i] \oplus O(y'[n] + i)$, for $1 \le i \le n - 1$
  4) Return $x[1]...x[n - 1]$
- *Encryption query for* $x[1]...x[n - 1]$
  1) Pick random $y'[n] \in \{0,1\}^l$
  2) Compute $y'[i] = x[i] \oplus O(y'[n] + i)$, for $1 \le i \le n - 1$
  3) Compute $t = y'[1] \oplus ... \oplus y'[n]$
  4) Compute $y[i] = y'[i] \oplus t$, for $1 \le i \le n$
  5) Return $y[1]...y[n]$

When A outputs two messages $x_1[1]...x_1[n-1]$ and $x_2[1]...x_2[n - 1]$, B picks $b \in \{0,1\}$ 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 $1 \le i \le n-1$
3) Compute $t = y_b'[1] \oplus \ldots \oplus y_b'[n]$
4) Compute $y_b[i] = y_b'[i] \oplus t$, for $1 \le i \le n$

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

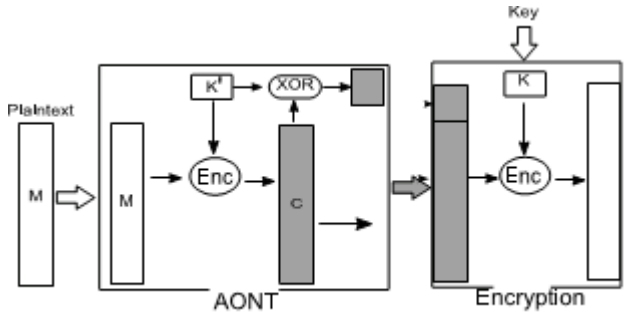

Fig 2: Current AON encryption schemes require a pre-processing round of block cipher encryption for the AONT, followed by another round of block cipher encryption.

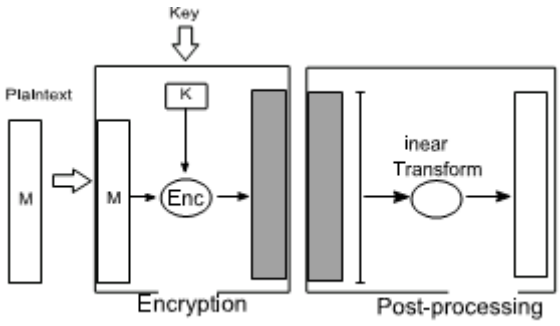

Fig 3: On the other hand, Bastion first encrypts the data with one round of block cipher encryption,

5.COMPARISON TO EXISTING SCHEMES

In what follows, we briefly overview several encryption modes and argue about their security (according to Definitions 1 and 2) and performance when compared to Bastion.

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.

5.1Performance Comparison

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 use the ECB encryption mode), requires additional n block cipher operations, and yields an AON encryption that is *ind* secure[7] and $(n-1)$*CAKE* secure. Encrypt-then-secretshareis *ind* secure and $(n-1)$*CAKE* secure. It requires $(n-1)$ block cipher operations and n XOR operations if additive secret sharing is used. However 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

6.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 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 and Desai. For baseline comparison, we include in our evaluation the CTR encryption mode and the AONTs due to Rivest and

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 incurs a negligible overhead when compared to existing semantically secure encryption modes (e.g., the CTR encryption mode) that are only 1*CAKE* secure.

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 2 and 3 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).

7.RELATED WORK

To the best of our knowledge, this is the first work that addresses the problem of securing data stored in multicloud 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 "sharedkey deniable encryption"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.

3. CONCLUSIONS

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 analyzed the security of Bastion and evaluated its performance in realistic settings. Bastion considerably 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 doublyiterated, ideal ciphers," in *Advances in Cryptology (CRYPTO)*, 1998, pp. 390–407.

[4] C. Basescu, C. Cachin, I. Eyal, R. Haas, and M. Vukolic, "Robust Data Sharing with Key-value Stores," in *ACM SIGACTSIGOPS Symposium on Principles of Distributed Computing (PODC)*, 2011, pp. 221–222.

[5] A. Beimel, "Secret-sharing schemes: A survey," in *International 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-ofclouds," 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 Allor-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. Kilian, 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 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 Deniable 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 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.