# Securing Cloud Data in the New Attacker Model

Ghassan O. Karame[1], Claudio Soriente[2], Krzysztof Lichota[3], Srdjan Čapkun[2]

[1] NEC Laboratories Europe, Germany
[2] ETH Zurich, Switzerland
[3] 9livesdata, Poland

**Abstract.** The world just witnessed the surge of a new and powerful attacker, which was able to coerce operators and acquire the necessary keys to break the privacy of users. Once the encryption key is exposed, the only viable measure to preserve data confidentiality is to limit the adversary's access to the ciphertext. This may be achieved, for example, using multi-cloud storage systems. These systems spread data across multiple servers in different administrative domains, to cater for availability and fault tolerance. If the adversary can only compromise a subset of these domains, multi-cloud storage systems may prevent the adversary from accessing the entire ciphertext. However, if data is encrypted using existing encryption schemes, spreading the ciphertext on multiple servers does not entirely solve the problem since an adversary which has the encryption key, can still compromise single servers and decrypt the ciphertext stored therein.
In this paper, we leverage multi-cloud storage systems to provide data confidentiality against an adversary which has access to the encryption key, and can compromise a large fraction of the storage servers. For this purpose, we first introduce a novel security definition that captures data confidentiality in the new adversarial model. We then propose Bastion, a primitive that is secure according to our definition and, therefore, guarantees data confidentiality even when the encryption key is exposed, as long as the adversary cannot compromise *all* storage servers. We analyze the security of Bastion, and we evaluate its performance by means of a prototype implementation. Our results show that Bastion incurs less than 5% overhead compared to existing semantically secure encryption modes. We also discuss practical insights with respect to the integration of Bastion in commercial multi-cloud storage systems.

## 1 Introduction

The year 2013 witnessed the introduction of a new, and extremely powerful attacker model. Notably, the world became aware of a massive surveillance program which mined data from operators and ISPs, and performed illegal taps on digital communication channels [27]. This surveillance program was not hindered by the various security measures deployed within the targeted services. For instance, although these services relied on secure encryption mechanisms, the necessary keying material was acquired, e.g., by means of backdoors, bribe, or coercion.

In addition to the public and governmental outrage, another immediate reaction from the industry was an even larger apprehension to use third-party services, and in particular cloud services [15]. If the encryption key is exposed, the only viable countermeasure 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, this countermeasure does not entirely solve the problem. Even if the data is encrypted and dispersed across different administrative domains, an adversary equipped with the appropriate keying material can compromise a single server and decrypt ciphertext blocks stored therein.

In this paper, we leverage multi-cloud storage systems [2, 4, 13, 14, 26, 28] to provide data confidentiality against an adversary which knows the encryption key, and can compromise a large fraction of the storage servers. The adversary can acquire the keys either by exploiting flaws or backdoors in the key-generation software [27], 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 cryptographic solutions, including those that protect the keys by means of secret-sharing (since the keys are leaked at generation time).

We start by proposing a novel game-based definition which captures the capabilities of the new adversarial model. Our definition, called $(n-\lambda)$ key-exposure or $(n-\lambda)$*ke*, models an adversary that has access to the encryption key and requires that no plaintext block can be recovered, as long as the adversary has access to at most $(n-\lambda)$ ciphertext blocks, where $n$ is the total number of ciphertext blocks.

The definition of $(n-\lambda)$*ke* shares similarities with the notion of all-or-nothing transformations (AONT) [5,9,23]. An AONT is a block-wise transform (based on block ciphers) that cannot be inverted unless all the output blocks are known. AONT is not an encryption by itself, but can be used as a pre-processing step before encrypting the data with a block cipher, to ensure that the plaintext can be decrypted by the key holder, only if all of the ciphertext blocks are available. The resulting encryption—called AON encryption—was mainly intended to slow down brute-force attacks on the encryption key. However, AON encryption can also preserve the confidentiality of the plaintext in case the encryption key is exposed, as long as the adversary has access up to all but one ciphertext blocks. In other words, given a ciphertext of $n$ blocks, AON encryption is $(n-1)$*ke* secure. Existing AON encryption modes, however, require *at least* two rounds of block cipher encryptions on the data: one pre-processing round to create the AONT, followed by another round for the actual encryption. This results in considerable—often unacceptable—overhead to encrypt and decrypt large files.

In this work, we propose $\mathrm{Bastion}$, an efficient primitive that requires only one round of block cipher encryption, followed by a linear transformation. $\mathrm{Bastion}$ is $(n-2)$*ke* secure, i.e., it ensures that plaintext data cannot be recovered as long as the adversary has access up to all but *two* ciphertext blocks, even when the encryption key is exposed. As such, $\mathrm{Bastion}$ relaxes the notion of all-or-nothing at the benefit of improved performance. This is reasonable since, in a multi-cloud storage system, each server is likely to store more than one ciphertext block. For example, if each server stores at least two ciphertext blocks, a $(n-2)$*ke* secure scheme clearly preserves data confidentiality unless *all* servers are compromised, even when the adversary has access to the encryption key. We analyze the security of $\mathrm{Bastion}$, and we compare its performance in a realistic implementation setup with a number of existing encryption schemes. Our results show that $\mathrm{Bastion}$ only incurs a negligible performance deterioration (less than 5%) when compared to symmetric encryption schemes (e.g., the $\mathrm{CTR}$ encryption mode), and con-

siderably improves the performance (by more than 50%) of existing AON encryption schemes [9, 23]. Finally, we discuss practical insights with respect to the possible integration of Bastion in commercial dispersed storage systems.

The remainder of the paper is organized as follows. In Section 2, we define our notation and building blocks. In Section 3, we describe our model and introduce a novel security definition modeling the new attacker. In Section 4, we present our scheme, Bastion, and analyze its security. In Section 5, we implement and evaluate Bastion in realistic settings. In Section 6, we 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.

## 2 Preliminaries

We adapt the notation of [9] 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 $F \in B(k,l)$, we denote $F^{-1} \in BC(k,l)$ as $F^{-1}(a,y)$ or as $F_a^{-1}(y)$, for $a \in \{0,1\}^k$.

***Encryption modes.*** An encryption mode based on a block cipher $F/F^{-1}$ is given by a triplet of algorithms $\prod = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ where:

- $\mathcal{K}$ 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 $F_a^{-1}$.
- $\mathcal{E}$ 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 $y$.
- $\mathcal{D}$ 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 $a \leftarrow \mathcal{K}(1^k)$, for any message $x \in \{0,1\}^*$, and for any $y \leftarrow \mathcal{E}^{F_a, F_a^{-1}}(x)$, we have $x \leftarrow \mathcal{D}^{F_a, F_a^{-1}}(y)$.

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

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

In the *ind* experiment, the adversary has unrestricted oracle access to $\mathcal{E}^{F_a, F_a^{-1}}$ during the "find" stage. At this point, $\mathcal{A}$ 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:

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

**Definition 1.** *An encryption mode $\prod = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ is ind-secure if for any polynomial-time adversary $\mathcal{A}$, we have $\mathrm{Adv}^{ind}_{\prod}(\mathcal{A}) \leq \epsilon$, where $\epsilon$ is a negligible function in the security parameter.*

***All or Nothing Transformations.*** An All or Nothing Transformation (AONT) is an efficiently computable transformation that maps sequences of input blocks to sequences of output blocks with the following properties: *(i)* given all the output blocks, the transformation can be efficiently inverted (i.e., the original input can be computed), 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 polynomial-time algorithms $\prod = (\mathbb{E}, \mathbb{D})$ where:

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

The literature comprises a number of security definitions for AONT (e.g., [5,9,23]). In this paper, we rely on the definition of [9] which uses the *aont* experiment below. This definition specifies a block length $l$ such that the pseudo-ciphertext $y$ can be written as $y = y[1] \ldots y[n]$, where $|y[i]| = l$ and $n \geq 1$.

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

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 it is allowed all but one encoded blocks. The advantage of $\mathcal{A}$ in the *aont* experiment is given by:

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

**Definition 2.** *An All-or-Nothing Transformation $\prod = \mathbb{E}, \mathbb{D}$ is aont-secure if for any polynomial-time adversary $\mathcal{A}$, we have $\mathrm{Adv}_{\prod}^{aont}(\mathcal{A}) \leq \epsilon$, where $\epsilon$ is a negligible function in the security parameter.*

***Proposed AONTs.*** Rivest [23] 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 $x[1] \ldots x[m]$, the package transform outputs $y[1] \ldots y[n]$, with $n = m+1$, where

$$y[i] = x[i] \oplus F_K(i),\ 1 \leq i \leq n-1,$$
$$y[n] = K \bigoplus_{i=1}^{n-1} F_{K_0}(y[i] \oplus i).$$

Desai [9] proposed a faster version where the block cipher round which uses $K_0$ is skipped and the last output block is set to $y[n] = K \bigoplus_{i=1}^{n-1} y[i]$. Both AONTs are secure according to Definition 2 [9].

*Remark 1.* Although the majority of proposed AONTs are based on block ciphers (e.g., [9, 23]), an AONT is not an encryption scheme. In particular, there is no secret-key information associated with an AONT. That is, given all the output blocks of the AONT, any party can recover the original input without knowledge of any secret.

## 3 $(n - \lambda)$ Key-Exposure Security

In this section, we detail our model, and we introduce the notion of $(n-\lambda)$ key-exposure security.

### 3.1 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 [2, 4, 28] with leading cloud storage providers such as EMC, IBM, and Microsoft, offering products for multi-cloud systems [13, 14, 26].

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 [18] which exports two operations:

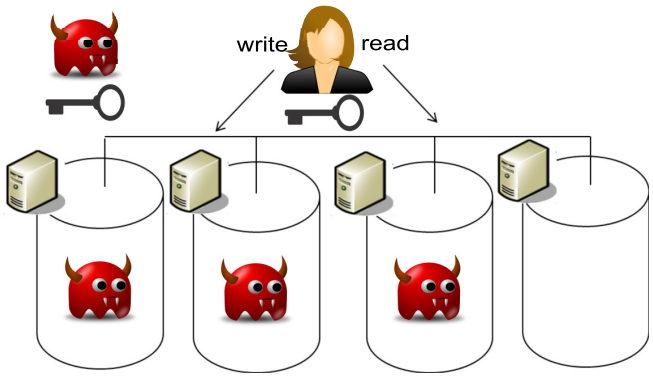



**Fig. 1.** The new attacker model. We assume a powerful adversary which can acquire all the cryptographic secret material, and can compromise a large fraction (up to all but one) of the storage servers.

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

read() The read routine fetches the stored value $v$ from the servers. For each $j \in [1 \ldots s]$, piece $v_j$ is downloaded from server $S_j$ and all pieces are combined into $v$. We assume that the initial value of the storage is a special value $\perp$, which is not a valid input value for a write operation.

We assume that each user encrypts the data before invoking the write() routine. To store file $f$, a user generates an encryption key $K$, encrypts $f$ into $v \leftarrow \mathcal{E}_K(f)$, and invokes write($v$). Key $K$ is stored at the user's machine. At a later time, the user invokes read() to fetch $v$ and runs $f \leftarrow \mathcal{D}_K(v)$ to recover $f$.

***The New Attacker Model.*** We assume the existence of a computationally-bounded adversary $\mathcal{A}$ which *can acquire all cryptographic keys.* The adversary may do so either *(i)* by leveraging flaws or backdoors in the key-generation software [27], *(ii)* by compromising the cloud infrastructure, or *(iii)* by compromising the user-device that stores the keys.

Since multi-cloud systems spread data across different domains, we assume that the adversary can compromise a large fraction (up to all but one) of the storage servers (cf. Figure 1). We model this adversary assuming that it has access to all but $\lambda$ ciphertext blocks, where $\lambda$ is a parameter of the encryption mode. While confidentiality with $\lambda = 0$ is clearly not achievable[4], our goal is to design encryption modes where $\lambda$ is as close as possible to 1.

Note that if the adversary additionally learns the user's credentials, logs into the storage servers, and acquires all the ciphertext blocks, then no cryptographic mechanism can preserve data confidentiality. However, we stress that compromising the encryption key does not necessarily imply the compromise of the user access credentials.

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

For example, encryption can occur on a specific-purpose device (e.g., [7]), and the key can be leaked, e.g., by the manufacturer; in this scenario, the user credentials to access the cloud servers are clearly not compromised.

### 3.2 The Notion of $(n - \lambda)$ Key-Exposure Security

Existing security notions for encryption modes (e.g., Definition 1) 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. We argue that the surge of "the new attacker model"—where the attacker has unrestricted access to the encryption key—clearly motivates the re-design of existing security notions. We capture the notion of confidentiality in the new attacker model with a new game-based definition for encryption modes, that we call $(n - \lambda)$ key-exposure or $(n - \lambda)ke$. Similar to [9], $(n - \lambda)ke$ specifies a block length $l$ such that a ciphertext $y$ can be written as $y = y[1] \dots y[n]$ where $|y[i]| = l$ and $n \geq 1$.

$$
\begin{array}{l}
\mathbf{Exp}_{\prod}^{(n-\lambda)ke}(A, b) \\
\quad a \leftarrow \mathcal{K}(1^k) \\
\quad x_0, x_1, state \leftarrow A^{\mathcal{E}^{F_a, F_a^{-1}}}(find) \\
\quad y_b \leftarrow \mathcal{E}^{F_a, F_a^{-1}}(x_b) \\
\quad b' \leftarrow A^{Y_b, \mathcal{E}^{F_a, F_a^{-1}}}(guess, state)
\end{array}
$$

The adversary has unrestricted access to $\mathcal{E}^{F_a, F_a^{-1}}$ in both the "find" and "guess" stages. On input $j$, the oracle $Y_b$ returns $y_b[j]$ and accepts up to $n - \lambda$ queries. On the one hand, unrestricted oracle access to $\mathcal{E}^{F_a, F_a^{-1}}$ 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 $\lambda$ ciphertext blocks. This is the case when, for example, $\lambda$ ciphertext blocks are stored in a server that the adversary cannot compromise (cf. Figure 1). The advantage of the adversary is defined as:

$$Adv_{\prod}^{(n-\lambda)ke}(A) = Pr[\mathbf{Exp}_{\prod}^{(n-\lambda)ke}(A, 1) = 1] - Pr[\mathbf{Exp}_{\prod}^{(n-\lambda)ke}(A, 0) = 1]$$

**Definition 3.** *An encryption mode $\prod = \mathcal{K}, \mathcal{E}, \mathcal{D}$ is* $(n-\lambda)$ke*-secure if for any polynomial-time adversary $\mathcal{A}$, we have $\mathrm{Adv}_{\prod}^{(n-\lambda)\mathrm{ke}}(\mathcal{A}) \leq \epsilon$, where $\epsilon$ is a negligible function in the security parameter.*

Definition $(n - \lambda)ke$ resembles Definition 2 but has two fundamental differences. First, $(n - \lambda)ke$ refers to a keyed scheme and gives the adversary unrestricted access to the encryption/decryption oracles. Second, $(n-\lambda)ke$ relaxes the notion of all-or-nothing and parameterizes the number of ciphertext blocks that are given to the adversary. As we will show later, 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 $(n-\lambda)ke$ does not consider confidentiality against "traditional" adversaries (i.e., adversaries which do not know the encryption key). Therefore, we seek for an encryption mode $\prod$ with the following properties:

1. $\prod$ must be *ind*-secure against an adversary which does not know the encryption key (cf. Definition 1).
2. $\prod$ must be $(n-\lambda)ke$-secure against an adversary which knows the encryption key (cf. Definition 3).

**Possible Solutions:** In what follows, we briefly overview several encryption modes and argue about their security according to Definition 1 and to Definition 3.

***CPA-encryption modes.*** Traditional CPA-encryption modes, such as the $\mathrm{CTR}$ mode, provide *ind*-security but are only $1ke$ secure. That is, an adversary equipped with the encryption key must only fetch two ciphertext blocks to break data confidentiality.[5]

***CPA-encryption and secret-sharing.*** Another option is to rely on the combination of CPA-secure encryption modes and secret-sharing. In particular, a file $f$ is first encrypted using a CPA-secure encryption mode. The resulting ciphertext is then secret-shared in $n$ shares with the reconstruction threshold set to $(n-1)$. This scheme is clearly $(n-1)ke$ secure and is also secure according to Definition 1. However, the resulting ciphertext is $n$ times larger than the size of $f$, which makes it impractical for storing large files. We also point out that secret-sharing the encryption key does not provide better security against the adversary we consider. This is because, in our model, the adversary has access to the encryption key at generation time, and before the key shares are distributed across the servers.

***AON encryption.*** Another alternative is to pre-process a message with an AONT and then encrypt its output with a CPA-secure encryption mode.[6] This paradigm is referred to in the literature as AON encryption and was first suggested by [23]. Existing AON encryption schemes require at least two rounds of block cipher encryption with two different keys: the first round is the actual AONT that embeds the first encryption key in the pseudo-ciphertext (cf. Section 2); a second round uses another encryption key that is kept secret to guarantee CPA-security. AON encryption increases the storage size by a single block. However, two encryption rounds constitute a considerable overhead when encrypting and decrypting large files.

Clearly, all existing solutions are either not satisfactory in terms of security or incur a large overhead and may not be suitable to store large files in a multi-cloud storage system. In what follows, we introduce our solution, $\mathrm{Bastion}$, which considerably improves the performance of existing solutions.

[5] We assume that the CTR encryption routine uses a random IV that is incremented at every block encryption.

[6] Recall that AONT alone, is not an encryption scheme and does not require the decryptor to have any key material.

**Fig. 2.** Encryption in Bastion.

**procedure** $\mathcal{E}(K, x = x[1] \dots x[m])$
  $n = m + 1$
  $y'[n] \leftarrow \{0,1\}^l$ ▷ y'[n] is the IV for CTR
  **for** $i = 1 \dots n-1$ **do**
    $y'[i] = x[i] \oplus F_K(y'[n] + i)$
  **end for**
  $t = y'[1]$
  **for** $i = 2 \dots n$ **do**
    $t = t \oplus y'[i]$
  **end for**
  **for** $i = 1 \dots n$ **do**
    $y[i] = y'[i] \oplus t$
  **end for**
  **return** $y$ ▷ $y = y[1] \dots y[n]$
**end procedure**

**Fig. 3.** Decryption in Bastion.

**procedure** $\mathcal{D}(K, y = y[0] \dots y[n])$
  $t = y[n]$
  **for** $i = 1 \dots n-1$ **do**
    $t = t \oplus y[i]$
  **end for**
  **for** $i = 1 \dots n$ **do**
    $y'[i] = y[i] \oplus t$
  **end for**
  **for** $i = 1 \dots n-1$ **do**
    $x[i] = y'[i] \oplus F_K^{-1}(y'[n] + i)$
  **end for**
  **return** $x$ ▷ $x = x[1] \dots x[n-1]$
**end procedure**

## 4 Bastion: An Efficient $(n-2)$*ke* Primitive

In this section, we present an efficient $(n-2)$*ke* secure encryption scheme, called Bastion. 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.

### 4.1 Description

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 $x[1], \dots, x[m]$, where $m$ is odd[7] such that each block has size $l$.[8] The set of input blocks is encrypted under key $K$, resulting in ciphertext $y' = y'[1], \dots, y'[m+1]$, where $y'[m+1]$ is randomly chosen from $\{0,1\}^l$.

Next, Bastion applies a linear transformation to $y'$, inspired by [25]. 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.[9] 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 $i = 1 \dots, n$.

Given key $K$, inverting Bastion entails computing $y' = y \cdot A^{-1}$ and decrypting $y'$ using $K$.[10]

[7] This requirement is essential for the correctness of the subsequent linear transformation on the ciphertext blocks. That is, if $m$ is not odd, then the transformation is not invertible.

[8] $l$ is the block size of the particular block cipher used.

[9] $0^l$ and $1^l$ denote a bitstring of $l$ zeros and a bitstream of $l$ ones, respectively.

[10] Matrix $A$ is invertible and $A = A^{-1}$.

The pseudocode of the encryption and decryption algorithms of Bastion are shown in Figure 2 and Figure 3, respectively. We efficiently implement the linear transform using $2n$ XOR operations by computing:

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

Note that $y'[1] \ldots y'[n]$ computed up to line 6 is the output of the CTR encryption mode, where $y'[n]$ is the initialization vector. Similar to CTR encryption mode, the final output of Bastion is one block larger than the original input.

*Remark 2.* We point out that Bastion is not restricted to the CTR mode and can be instantiated with other *ind*-secure block cipher encryption modes (e.g., CBC, OFB).

### 4.2 Security Analysis

In what follows, we analyze the security of Bastion.

**Lemma 1.** *Bastion is* ind*-secure.*

**Proof Sketch 1.** Bastion uses an ind-secure encryption mode (CTR mode) to encrypt a message, and then applies a linear transformation on the ciphertext blocks. It is straightforward to conclude that Bastion is $ind$-secure.

**Lemma 2.** *Given any $n-2$ blocks of $y[1] \ldots y[n]$ as output by* Bastion*, it is infeasible to compute any $y'[i]$, for $1 \leq i \leq n$.*

**Proof Sketch 2.** Let $y = y[1], \ldots, y[n] \leftarrow \mathcal{E}(K, x = x[1] \ldots x[m])$. 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 $1 \leq i \leq n$. 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 $1 \leq i \leq n$.

**Lemma 3.** *Bastion is* $(n-2)$ke *secure.*

**Proof Sketch 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 an adversary $\mathcal{A}$ which has non-negligible advantage in the $(n-\lambda)ke$ experiment with $\lambda = 2$, we can construct an adversary $\mathcal{B}$ which has non-negligible advantage in distinguishing between a true random permutation and a pseudo-random permutation. Appendix A provides the details of the proof.

*Remark 3.* Bastion is not $(n-1)ke$ 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 $(n-1)ke$ game. Recall that our security definition allows the adversary to learn the encryption key.

### 4.3 Performance Analysis

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 2*ke* 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.[11] Encrypting the output of Rivest AONT or Desai AONT with a standard encryption mode (both [9] and [23] use the ECB encryption mode), requires additional $n$ block cipher operations, and yields an AON encryption that is *ind*-secure[12] and $(n - 1)$*ke* secure. Secret-sharing encryption (cf. 3.2) is *ind*-secure and $(n - 1)$*ke* 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 $(n-2)$*ke* security with only $(n - 1)$ block cipher operations and $(3n - 1)$ XOR operations $(n - 1)$ XOR operations for the CTR encryption and $2n$ XOR operations for the linear transformation).

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 5, we confirm the superior performance of Bastion by means of implementation.

## 5 Implementation and Evaluation

In this section, we describe and evaluate a prototype implementation modeling a read-write storage system based on Bastion.

### 5.1 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 2) using both AES128 and Rijandael256, 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 [23] and Desai [9]. For baseline comparison, we include in our evaluation the CTR encryption

[11] Note that removing the pseudo-ciphertext block where the key is embedded would not achieve *ind*-security, because the transformation is not randomized.

[12] 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 is fixed.

**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) | ***ind*-secure** | $(n - \lambda)$***ke* secure** |
|---|---|---|---|---|
| **CTR encryption mode** | $n - 1$ b.c.<br>$n - 1$ XOR | $n$ | yes | $1ke$ |
| **Rivest AONT** [23] | $2(n - 1)$ b.c.<br>$3(n - 1)$ XOR | $n$ | no | N/A |
| **Desai AONT** [9] | $n - 1$ b.c.<br>$2(n - 1)$ XOR | $n$ | no | N/A |
| **Rivest AON Encryption** [23] | $3n - 2$ b.c.<br>$3(n - 1)$ XOR | $n$ | yes | $(n - 1)ke$ |
| **Desai AON Encryption** [9] | $2n - 1$ b.c.<br>$2(n - 1)$ XOR | $n$ | yes | $(n - 1)ke$ |
| **CTR and Secret-sharing** | $n - 1$ b.c.<br>$2n - 1$ XOR | $n^2$ | yes | $(n - 1)ke$ |
| **Bastion** | $n - 1$ b.c.<br>$3n - 1$ XOR | $n$ | yes | $(n - 2)ke$ |

mode and the AONTs due to Rivest [23] and Desai [9], which are used in existing dispersed storage systems, e.g., Cleversafe [22]. We do not evaluate the performance of secret-sharing 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 CPU 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 with respect to various file sizes and block sizes. For each data point, we report the average of 30 independent runs. Due to their small width (in the order of 0.1), we do not show the corresponding 95% confidence intervals.

### 5.2 Evaluation Results

Our evaluation results are reported in Figure 4 and Figure 5. Both figures show that Bastion considerably improves (by more than 50%) the performance of existing $(n - 1)$*ke* 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 $2$*ke* secure.

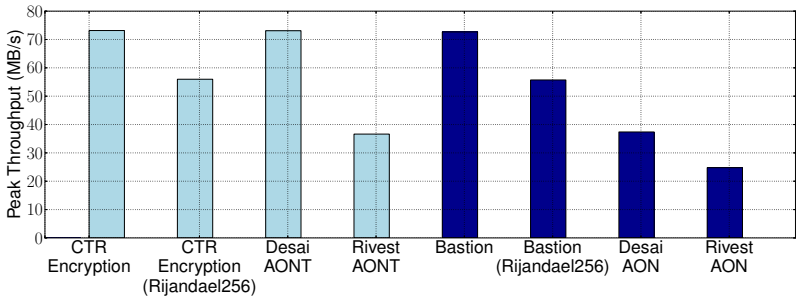



**Fig. 4.** Peak throughput comparison. Unless otherwise specified, the underlying block cipher is AES128. Each data point is averaged over 30 runs. Histograms in dark blue depict encryption modes which offer comparable security to Bastion. Light blue histograms refer to encryption/encoding modes where individual ciphertext blocks can be inverted when the key is exposed.

In Figure 4, 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 $(n-1)$*ke* 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 5(a) and 5(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).

## 6 Practical Insights

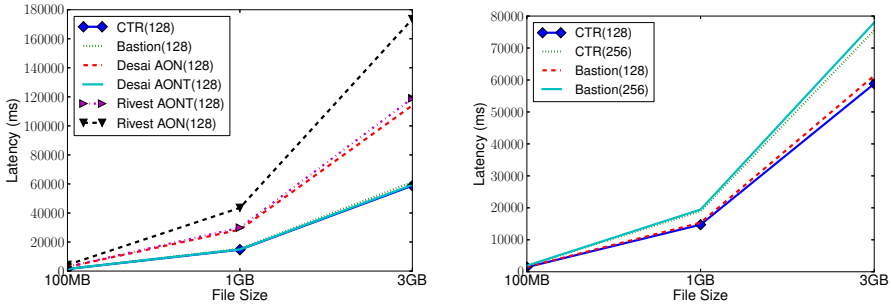
In this section, we discuss practical insights with respect to the integration of Bastion within existing dispersed storage systems.

### 6.1 Bastion in Dispersed Storage Systems

Bastion is $(n-2)$*ke* secure according to Definition 3. However, in a multi-cloud storage system, since each server stores at least two ciphertext blocks (i.e., $2l$ bits of data)[13], a $(n-2)$*ke* secure scheme like 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., [21]) to provide data confidentiality and fault tolerance.

[13] For example, if a 10 MB file is distributed across 10 servers, each server would store around 1 MB of data.




(a) Latency of encryption/encoding for different file sizes.

(b) Latency of encryption/encoding for different block sizes of the underlying block cipher.

**Fig. 5.** Performance evaluation of Bastion. Each data point in is averaged over 30 independent runs. Unless otherwise specified, the underlying block cipher is AES-128. CTR(256) and Bastion(256) denote the CTR encryption mode and Bastion encryption routine, respectively, instantiated with Rijandael256.

Recall that information dispersal algorithms (IDA), parameterized with $t_1, t_2$ (where $t_1 \leq t_2$), encode data into $t_2$ symbols such that the original data can be recovered from any $t_1$ encoded symbols.

In our multi-cloud storage system (cf. Section 3.1), a file $f$ of size $ml$-bits results in a ciphertext of $n = m + 1$ blocks (of $l$ bits each) when encrypted using Bastion.[14] The ciphertext is then fed to the IDA encoding routine with parameters $t_1 \leq \frac{n}{2}$ and $t_2 = s > t_1$, where $s$ is the number of available servers. By setting $t_1 \leq \frac{n}{2}$, we ensure that each symbol encodes 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 at least $t_1$ distinct symbols from any subset of the servers and decoding them via the IDA decoding routine. The resulting ciphertext can be decrypted using Bastion to recover the original file $f$.

By doing so, data confidentiality is preserved in the new attacker model unless $t_1$ servers are compromised (and their stored data is acquired). Furthermore, data availability is guaranteed in spite of $s - t_1$ server failures.

### 6.2 Deployment within HYDRAstor

We now discuss the integration of a prototype implementation of Bastion within the HYDRAstor grid storage system [11,20]. HYDRAstor is a commercial secondary storage solution for enterprises, which consists of a back-end architectured as a grid of storage nodes built around a distributed hash table. The front-end of HYDRAstor consists of a layer of access nodes which are scaled for performance. HYDRAstor relies on system-wide deduplication, and supports online extensions and upgrades. It also tolerates multiple disk, node and network failures, rebuilds the data automatically after

[14] Since Bastion is $(n-2)ke$ secure, an adversary equipped with the encryption key, must collect at least $n - 1$ out of $n$ ciphertext blocks to break data confidentiality.

failures, and informs users about recoverability of the deposited data. 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 [11].

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 (cf. Section 6.1).

In order to better assess the performance impact of Bastion in HYDRAstor, we evaluated the performance of Bastion in the newest generation HYDRAstor 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 (non-duplicate) to remove the effect of data deduplication. Our results show that the write bandwidth was not affected by the integration of Bastion. On the other hand, 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 in order to strengthen the security of these systems in the new attacker model, without affecting performance.

## 7 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, all-or-nothing transformations, secret-sharing techniques, and leakage-resilient cryptography.

**Deniable Encryption:** Our work shares similarities with the notion of "deniable encryption" [6,12,16]. An encryption scheme is "deniable" if—when coerced to reveal the encryption key—the legitimate owner can make the ciphertext "look like" the encryption of a plaintext different from the original one, thus keeping the original plaintext private. Deniable encryption is typically achieved by constructing a ciphertext which can be decrypted into a number of plausible plaintexts, under different keying material. Deniable encryption therefore aims to deceive an adversary which does not know all the keying material but, e.g., can acquire a subset of the keys. Unlike deniable encryption, Bastion is secure against an adversary which knows *all* the necessary keying material.

**All or Nothing Transformations:** All-or-nothing transformations (AONTs) were first introduced in [23] and later studied in [5, 9]. The majority of AONTs leverage a secret key that is embedded 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. To construct an encryption scheme which is secure in the new attacker model, one alternative is to preprocess a message with an AONT and encrypt its output with a CPA-secure encryption mode. This paradigm is referred to in the literature as AON encryption [5, 9, 23]. Existing AON encryption schemes require at least two rounds of encryption, while Bastion

achieves a comparable level of security with considerably better performance (by more than 50%). Resch et al. [22] combine AONT and information dispersal to provide both fault-tolerance and data secrecy, in the context of distributed storage systems. In [22], however, an adversary which knows the encryption key can decrypt data stored on single servers.

**Secret Sharing:** Secret sharing schemes [3] 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 [8, 24], 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 [21] proposed an information dispersal algorithm with smaller overhead than the one of [24], however, this proposal does not provide any security guarantees when a small number of shares (less than the reconstruction threshold) are available. Krawczyk [17] proposed to combine both Shamir's [24] and Rabin's [21] approaches; in [17] a file is first encrypted using AES and then dispersed using the scheme in [21], while the encryption key is shared using the scheme in [24]. 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 stand against adversary which learns partial information about the secret state of a system, e.g., through side-channels [19]. Different models allow to reason about the "leaks" of real implementations of cryptographic primitives [1, 10, 19]. 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.

## 8 Conclusion

In this paper, we addressed the problem of securing data outsourced to the cloud in the new attacker model, in which the adversary can acquire the encryption keys (e.g., using bribe, coercion). For this purpose, we introduced a novel security definition that captures data confidentiality against the new adversary. We then proposed Bastion, a primitive which ensures the confidentiality of encrypted data even when the adversary has the encryption key, and all but *two* ciphertext 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 considerably improves (by more than 50%) the performance of existing primitives which offer comparable security in the new attacker model, 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 commercial dispersed storage systems.

## References


1. A. Akavia, S. Goldwasser, and V. Vaikuntanathan. Simultaneous hardcore bits and cryptography against memory attacks. In *Theory of Cryptography Conference (TCC)*, pages 474–495, 2009.
2. C. Basescu, C. Cachin, I. Eyal, R. Haas, and M. Vukolic. Robust Data Sharing with Key-value Stores. In *Proceedings of the 30th Annual ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing*, PODC '11, pages 221–222, New York, NY, USA, 2011. ACM.
3. A. Beimel. Secret-sharing schemes: A survey. In *Third International Workshop on Coding and Cryptology (IWCC)*, pages 11–46, 2011.
4. A. Bessani, M. Correia, B. Quaresma, F. André, and P. Sousa. DepSky: Dependable and Secure Storage in a Cloud-of-clouds. In *Proceedings of the Sixth Conference on Computer Systems*, EuroSys '11, pages 31–46, New York, NY, USA, 2011. ACM.
5. V. Boyko. On the Security Properties of OAEP as an All-or-nothing Transform. In *Proceedings of CRYPTO*, pages 503–518, 1999.
6. R. Canetti, C. Dwork, M. Naor, and R. Ostrovsky. Deniable Encryption. In *Proceedings of CRYPTO*, 1997.
7. Cavalry. Encryption Engine Dongle. `http://www.cavalrystorage.com/en2010.aspx/`.
8. 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)*, pages 89–95, 1994.
9. A. Desai. The security of all-or-nothing encryption: Protecting against exhaustive key search. In *Advances in Cryptology (CRYPTO)*, pages 359–375, 2000.
10. Y. Dodis, Y. T. Kalai, and S. Lovett. On cryptography with auxiliary input. In *ACM Symposium on Theory of Computing (STOC)*, pages 621–630, 2009.
11. 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 *FAST'09: Proceedings of the 7th USENIX Conference on File and Storage Technologies*, pages 197–210, Berkeley, CA, USA, 2009. USENIX Association.
12. M. Duermuth and D. M. Freeman. Deniable Encryption with Negligible Detection Probability: An Interactive Construction. To Appear in EUROCRYPT, 2011. Available from `http://eprint.iacr.org/2011/066.pdf`.
13. EMC. Transform to a Hybrid Cloud. `http://www.emc.com/campaign/global/hybridcloud/index.htm`.
14. IBM. IBM Hybrid Cloud Solution. `http://www-01.ibm.com/software/tivoli/products/hybrid-cloud/`.
15. ITIF. How Much Will PRISM Cost the U.S. Cloud Computing Industry? `http://www2.itif.org/2013-cloud-computing-costs.pdf`.
16. M. Klonowski, P. Kubiak, and M. Kutylowski. Practical Deniable Encryption. In *Proceedings of SOFSEM: Theory and Practice of Computer Science*, 2008.
17. H. Krawczyk. Secret Sharing Made Short. In *International Conference on Advances in Cryptology*, 1993.
18. L. Lamport. On interprocess communication, 1985.
19. S. Micali and L. Reyzin. Physically observable cryptography (extended abstract). In *Theory of Cryptography Conference (TCC)*, 2004.
20. NEC Corporation. HYDRAstor Grid Storage System. `http://www.hydrastor.com`.
21. M. O. Rabin. Efficient Dispersal of Information for Security, Load Balancing, and Fault Tolerance. In *Journal of the Association for Computing Machinery*, pages 335–348, 1989.

22. 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)*, pages 191–202, 2011.
23. R. L. Rivest. All-or-Nothing Encryption and the Package Transform. In *International Workshop on Fast Software Encryption (FSE)*, pages 210–218, 1997.
24. A. Shamir. How to Share a Secret? In *Communications of the ACM*, pages 612–613, 1979.
25. D. R. Stinson. Something About All or Nothing (Transforms). In *Designs, Codes and Cryptography*, pages 133–138, 2001.
26. StorSimple. Learn about Cloud-integrated Storage. `http://www.storsimple.com/`.
27. Wikipedia. Edward Snowden. `http://en.wikipedia.org/wiki/Edward_Snowden#Disclosure`.
28. Z. Wu, M. Butkiewicz, D. Perkins, E. Katz-Bassett, and H. V. Madhyastha. SPANStore: Cost-effective Geo-replicated Storage Spanning Multiple Cloud Services. In *Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles*, SOSP '13, pages 292–308, New York, NY, USA, 2013. ACM.

## A Proof of Lemma 3

Given an adversary $\mathcal{A}$ such that $\mathrm{Adv}^{(n-2)ke}_{\mathrm{Bastion}}(\mathcal{A}) > \epsilon$, we show, in what follows, how to construct an adversary $\mathcal{B}$ which has non-negligible advantage in distinguishing between a true random permutation and a pseudo-random permutation.

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

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

When $\mathcal{A}$ outputs two messages $x_1[1] \dots x_1[n-1]$ and $x_2[1] \dots x_2[n-1]$, $\mathcal{B}$ 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 $1 \leq i \leq n-1$
3. Compute $t = y'_b[1] \oplus \dots \oplus y'_b[n]$
4. Compute $y_b[i] = y'_b[i] \oplus t$, for $1 \leq i \leq n$

At this point, $\mathcal{A}$ selects $(n-2)$ indexes $i_1, \dots i_{n-2}$ and $\mathcal{B}$ returns the corresponding $y_b[i_1], \dots, y_b[i_{n-2}]$. Encryption/decryption queries are answered as above. When $\mathcal{A}$ outputs its answer $b'$, $\mathcal{B}$ outputs 1 if $b = b'$, and 0 otherwise. It is straightforward to see that if $\mathcal{A}$ has advantage larger than negligible to guess $b$, then $\mathcal{B}$ has advantage larger

than negligible to distinguish a true random permutation from a pseudorandom one. Furthermore, the number of queries issued by $\mathcal{B}$ to its oracle amounts to the number of encryption and decryption queries issued by $\mathcal{A}$. Note that by Lemma 2, during the guess stage, $\mathcal{A}$ cannot issue a decryption query on the challenge ciphertext since with only $(n-2)$ blocks, finding the remaining blocks is infeasible.