Information leakage in cloud data warehouses

Mohammad Ahmadian, Member, IEEE, and Dan C. Marinescu, Senior Member, IEEE

Abstract—Information leakage is the inadvertent disclosure of sensitive information through correlation of records from several databases/collections of a cloud data warehouse. Malicious insiders pose a serious threat to cloud data security and this justifies the focus on information leakage due to rogue employees or to outsiders using the credentials of legitimate employees. The discussion in this paper is restricted to NoSQL databases with a flexible schema. Data encryption can reduce information leakage, but it is impractical to encrypt large databases and/or all fields of database documents. Encryption limits the operations that can be carried on the data in a database. It is thus, critical to distinguish sensitive documents in a data warehouse and concentrate on efforts to protect them. The capacity of a leakage channel introduced in this work quantifies the intuitively obvious means to trigger alarms when an insider attacker uses excessive computer resources to correlate information in multiple databases. The Sensitivity Analysis based on Data Sampling (SADS) introduced in this paper balances the trade-offs between higher efficiency in identifying the risks posed by information leakage and the accuracy of the results obtained by sampling very large collections of documents. The paper reports on experiments assessing the effectiveness of SADS and the use of selective disinformation to limit information leakage. Cloud services identifying sensitive records and reducing the risk of information leakage are also discussed.

Index Terms—Database as a Service, Information leakage, Capacity of a leakage channel, Sensitivity analysis, Approximate Query Processing, Biased Sampling, Cross-Correlation estimation.

1 INTRODUCTION

I
formation leakage is the inadvertent disclosure of sensitive information. A malicious insider with access to the information stored by a cloud data warehouse is able to infer sensitive information through multiple database searches and cross-correlations among databases. This new threat to cloud security has received little attention in the past. The impact of information leakage will most likely amplify as the volume of data stored on public clouds by many organizations is steadily increasing. Often oblivious to the dangers of information leakage many governmental agencies and enterprises transition to private and to hybrid clouds with the belief that in addition to lower cost a cloud offers enhanced security.

Nowadays virtually all Cloud Service Providers (CSPs) offer Database as a Service (DBaaS) [14]. It is predicted that DBaaS will enjoy a solid annual growth rate for the foreseeable future. CSPs guarantee availability and scalability of cloud services, but the data confidentiality poses significant challenges in the face of new threats.

Unauthorized access to confidential information and data theft top the list of concerns of individuals and organization who relinquish the physical control of their data to a CSP [13]. Some of the new threats emanate from insider attackers who have the ability to correlate information from multiple cloud databases. Sensitive information could be inferred using information from low-risk datasets hosted by the same cloud.

The discussion in this paper is restricted to NoSQL databases with a flexible schema. A NoSQL database is a collection of documents $D = d_1, \ldots, d_n$ and is also called a

- Mohammad Ahmadian and Dan C. Marinescu are with Department of Computer Science, University of Central Florida, Orlando, FL, 32816.
- E-mail: ahmadian, dcm}@cs.ucf.edu

Manuscript received August 28, 2017; revised March 16, 2018.
warehouses with a very large number of datasets is a daunting
task even when using samples of modest size taken from
different datasets. Fortunately, the computational effort
made by an attacker to gather sensitive information can be
exploited for his demise. Two cloud information leakage
prevention methods are proposed: (i) limit the ability of
an attacker to gather sensitive information by restricting
computing resources available for such attacks [25]; and (ii)
insert disinformation documents.

The first approach is based on a quantitative character-
ization of the capacity of a leakage channel discussed in
Section 3. Alarms triggered once pre-established thresholds
on the number of chained queries is exceeded serve as
deterrents for potential attackers and limit their ability to
collect sensitive information.

The second approach proposes the insertion of disinfor-
mation documents providing multiple values to an attribute
and misleads an attacker. The indiscriminate document
replication drastically increases the database size and,
implicitly, the query response time. The selective disinforma-
tion proposed in this paper uses sensitivity analysis to limit
the number of additional documents, as well as the other
inherent negative effects of the original method proposed in
[26].

The contributions of this paper are:
1) A survey of data encryption methods and their limita-
tions for preventing information leakage in cloud data
warehouses presented in Section 2.
2) The definition of the capacity of an information leakage
channel relating the level of information leakage with the
effort and the resources needed by a malicious insider,
discussed in Section 3.
3) The use of disinformation to limit the capacity of a
leakage channel, discussed in Section 4
4) The introduction of sensitivity analysis in the study
of information leakage through correlation of multiple
datasets in a database. An effective sensitivity analy-
sis method based on approximate query processing for
classifying documents in several sensitivity classes and
selective disinformation to limit information leakage are
discussed in Section 5.
5) A scalable leakage assessment and parameter extraction
algorithm for cloud data warehouses based on approxi-
mate query processing, discussed in Section 6.

The paper also suggests cloud services to assess the
likelihood of information leakage and limit the ability of
a malicious insider to discover sensitive information in cloud
data warehouses.

2 CLOUD DATA ENCRYPTION

Computer clouds are target-rich environments for malicious
individuals and criminal organizations [1], [2]. The impact
of traditional threats to all computer systems connected to
the Internet is amplified in the case of computer clouds
due to the vast amount of resources and the large user
population [3], [4], [5], [6], [7]. At the same time, new threats
that exploit cloud organization and services have emerged
[8], [9], [10], [11], [12].

Data encryption can be used for protection of sensitive
information, but cannot be used indiscriminately to protect
the very large volume of data stored on a cloud. Encryption
can be applied to a range of data granularity, from high
granularity of atomic data to low granularity of aggregated
data items. The higher the encryption granularity, the higher
the information leakage. For example, encryption of a single
attribute leaks how frequently the attribute is present in the
database records, while encryption of the entire document
and collection as a single unit leaks less information.

Cloud data can be in three states, at-rest, in-transit, or
in-process; a comprehensive data security mechanism must
protect data in any of the three states. Data encryption can
only protect cloud data while in storage, but arithmetic
operations with encrypted data are only theoretically fea-
sible at this time. It is therefore necessary to decrypt data
before processing and this process creates another window
of vulnerability and opens the door for information leakage.
It is feasible to query encrypted data as we shall see in
this section which starts with a review of cloud encryption
schemes and continues with an overview of two systems
used to process encrypted data.

Encryption Methods and Cryptosystems. Several encryp-
tion methods and cryptosystems used for cloud hosted
databases are discussed next.

I. Deterministic Encryption. This encryption scheme produces
the same ciphertext for an identical pair of plaintext and
key. For example, block ciphers in the Electronic Code
Book (ECB) mode, with a constant initialization vector are
deterministic. Deterministic encryption scheme preserves
equality; therefore, the frequency of encrypted data mirrors
the frequency of plaintext data and this information is
leaked to an attacker.

\[
\text{Eqn 1: } C_j = E_k(P_j); \quad P_j = D_k(C_j),
\]

Equation 1 describes the deterministic encryption and
decryption operation with \( E_k \) the encryption algorithm, \( D_k \)
the decryption algorithm, \( k \) the secret key, \( P_j \) a plaintext
data block, and \( C_j \) the ciphered data block.

II. Random Encryption. In this encryption scheme, a message
is coupled with a key \( k \) and a random Initial Vector (IV).
This scheme is non-deterministic, i.e., encryption of the
same message with the same key yields different ciphertext.
Random encryption schemes are semantically secure against
plaintext attacks. Equation 2 describes the encryption and
decryption of a block cipher in Cipher Block Chaining (CBC)
mode.

\[
\begin{align*}
\text{Eqn 2: } C_j &= E_k(P_j \oplus IV), \quad P_j = IV \oplus D_k(C_j), \\
C_j &= E_k(P_j \oplus C_{j-1}), \quad P_j = C_{j-1} \oplus D_k(C_j)
\end{align*}
\]

The Advanced Encryption Standard (AES) is one of the most
secure random encryption scheme. AES is a symmetric
block cipher algorithm with a key size of 128,192, or 256
bits and with a block size of 128 bits. Equation 3 shows
that random encryption function can be constructed from a
deterministic one by concatenation of a fixed length random
number \( r \) to each input.


$$E_k(x) = E'_k(x || r)$$  \hspace{1cm} (3)

where \( k \) is the encryption key, \( E' \) is a deterministic encryption, \( E_k(x) \) is a random encryption function, and \( r \) is a random number, illustrates this process.

**III. Fully Homomorphic Encryption (FHE)** allows computation to be done on encrypted data [33]. However, the full homomorphic encryption algorithm is very intricate and the overhead of computations with FH-encrypted data is several orders of magnitude larger than that of computations with the corresponding plaintext data. Another reason for FHE impracticality is that a query to an FH-encrypted database must be expressed as a circuit over an entire dataset.

**IV. Order-Preserving Encryption (OPE)** is a deterministic cryptosystem where ciphertext preserves the ordering of the plaintext data. Aggregate queries such as comparison, min, and max can be executed on OPE-encrypted datasets. Equation 4 shows the preservation of order relation of plaintext in the ciphertext with \( OPE_k \) the key-based OPE

$$\forall x, y | x, y \in \text{Data Domain} : x < y \implies OPE_k(x) < OPE_k(y).$$  \hspace{1cm} (4)

OPE offers less protection than FHE and leaks critical information about the plaintext data. The OPE algorithm introduced in [31] was used for a cloud database service [30]. The Modular Order-Preserving Encryption (MOPE) [32], an extension to the basic OPE claiming security improvements, also leaks information.

**V. Additive Homomorphic Encryption (AHOME)** is a partially homomorphic cryptosystem that allows a database server to conduct homomorphic addition and multiplication computations on ciphertext. An example of AHOME is Paillier’s cryptosystem [34]. The homomorphic addition is formulated as

$$D_k(E_k(m_1, r_1) . E_k(m_2, r_2) \mod n^2) = m_1 + m_2 \mod n$$  \hspace{1cm} (5)

where \( m_1, m_2 \in \mathbb{Z}_n \) are plaintext messages, \( r_1, r_2 \in \mathbb{Z}_n^* \) are randomly selected, and \( n \) is the product of two large primes, \( \mathbb{Z}_n \) and \( \mathbb{Z}_n^* \) are sets of integers.

Protection of data in-transit is discussed next. The communication channels with cloud databases can be secured using standard HTTP over the Secure Socket Layer (SSL) communication protocol. Most CSPs provide APIs for the web service enabling developers to use both the standard HTTP and the secure version of the HTTPS protocol. The security requirements for data in-transit can be fully satisfied using HTTPS for communication with a cloud. The endpoint authentication feature of SSL makes it possible to ensure that the clients are communicating with an authentic cloud server. The basic idea of maintaining confidentiality of data in at-rest and in-process states is to use a cryptosystem. However, providing the decryption key to the server is a confidentiality violation.

**Processing encrypted data.** Searchable encryption methods such as Oblivious RAM (ORAM) [19], [20] provide an acceptable level of security. However, the efficiency and high computational cost, as well as the excessive communication costs between the clients and the server make this method impractical [24]. Deterministic and OPE cryptosystems leak critical information such as frequency and order of the original data and enable attackers to extract sensitive information.

The following systems do not require modifications of database services; encrypted data is processed identically as plaintext data. Optimizations such as multi-layer indexing, caching, and file management operations are invariant whether applied to encrypted or plaintext databases. The first system, CryptDB, [17] is used to search encrypted SQL cloud databases. Inference attacks against CryptDB are discussed in [18].

The system to search encrypted NoSQL databases [16] involves a secure proxy to encrypt client queries and decrypt server query responses. The proxy ensures that an attacker could not access sensitive information. The process is completely transparent to the clients which are not involved in encryption/decryption operations.

It is impractical to encrypt all documents in a database or a large number of documents in multiple cloud databases. Moreover, encryption of all document fields restricts the range of database operations. Data encryption can be used to selectively protect sensitive information in a data warehouse however, it is seldom used for several reasons. First, it is not feasible at this time to support a full range of arithmetic and logic operations with encrypted data. Second, searching encrypted databases requires more complex software systems such as the ones discussed in [17] and [16].

This motivates the investigation of information leakage due to correlations among documents in plaintext or with partially encrypted fields in multiple databases of a data warehouse. Such correlations can only be carried out by individuals with quasi-unlimited access to the data as discussed in the next section.

**3 INFORMATION LEAKAGE DUE TO MALICIOUS INSIDERS ACCESS TO PLAINTEXT COLLECTIONS**

Malicious insiders could exploit information leakage from sensitive documents for a range of nefarious activities. Such attacks can be conducted by the employees and the contractors of large data centers with access to the software, the hardware, and the data. There is also the risk of an intruder gaining the same level of access using the credentials of a legitimate employee.

**Example.** The following example illustrates how correlations among multiple documents in several cloud collections allow an insider to infer sensitive information even when some sensitive documents are encrypted.

After buying an item from an online store a document, \( R_{m_0} \) of John’s sale including his name, address, phone number, and credit card information is stored in the cloud merchant’s collection, \( D_{m_0} \):

$$R_{m_0} = \{(\text{Name, John}), (\text{Addr, SW81}), (\text{Ph, 7654321}), (\text{Card, VISA_xgyzw|EXP_May20(COD_345)})\}.$$  \hspace{1cm} (6)

John’s dental documents are in the \( D_h \) collection on the same cloud. He is identified by a patient identification number, and credit card information is stored in the cloud merchant’s collection, \( D_{m_0} \):

$$R_{m_0} = \{(\text{Name, John}), (\text{Addr, SW81}), (\text{Ph, 7654321}), (\text{Card, VISA_xgyzw|EXP_May20(COD_345)})\}.$$  \hspace{1cm} (6)
This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSUSC.2018.2838520, IEEE Transactions on Sustainable Computing

number, PatId, stored as an encrypted document, $R_k^n$, to ensure anonymity of patient information:

$$R_k^n = \{(\text{Name}, \text{John}), (\text{PatId}, 987654)\} \quad (7)$$

After John visits an ophthalmologist a new document, $R_h^n$, containing the patient Id, age, sex, social security number, address, phone number and X-ray results is stored in the $D_h$ collection:

$$R_h^n = \{(\text{PatId}, 987654), (\text{Age}, 23), (\text{Sex}, M), (\text{SSN}, 333), (\text{Addr}, SW81), (\text{Ph}, 7654321), (\text{XRay}, \text{Results})\} \quad (8)$$

Encryption of the obviously sensitive document, $R_h^n$ is insufficient. Indeed, an insider with access to both collections, $D_m$ and $D_h$, can correlate the address and phone numbers in documents $R_m$ and $R_h$ and find John’s SSN and his credit card information, in spite of the attempt to protect John’s privacy by using the PatID instead of his name. This example illustrates the need for sensitivity analysis discussed in Section 5, to identify all fields of a document that need to be protected, in our case the address and the phone number.

The system model. We examine sensitive information leakage due to correlations among data in several NoSQL databases residing on the same cloud. We assume that data documents containing sensitive information consisting of $(\text{key}, \text{value})$ pairs, are distributed among several databases stored on the same cloud, and the attacker has access to all $N$ databases stored on the cloud.

In the general case, the intruder can attack a set of targets, $T = \{T_1, T_2, \ldots, T_q\}$. A target $T_i$, is the collection of documents scattered among the databases in a cloud hosted data warehouse containing sensitive information about one person, process, or document. The malicious insider knows at least one $(\text{key}, \text{value})$ pair for each target and has the potential to identify one $(\text{key}, \text{value})$ pair in every sensitive documents of each target.

3.1 The capacity of a leakage channel.

We propose a quantitative characterization of information leakage reflecting the attacker’s cost-benefits options. This measure correlates the amount of information leakage with the effort of the attacker; the larger the effort to access sensitive information, the higher the risk of detection.

The following analysis is based on several assumptions:

1) To avoid detection a malicious insider limits her effort to $n_i$ searches related to target $T_i$.

2) There are only $k_i < n_i$ documents with sensitive information related to $T_i$.

3) The attacker does not know $k_i$.

The $C^n_{T_i}(n_i, k_i)$, the capacity of an $[n_i, k_i]$-leakage channel relative to target $T_i$, is the probability of successful access to $r_1, r_2, \ldots, r_k$, sensitive documents relevant to target $T_i$ in $n_i$ searches.

$$C^n_{T_i}(n_i, k_i) = \prod_{j=1}^{n_i} p(n_j, k_j) \quad \text{with} \quad n = \sum_{j=1}^{n_i} n_j \quad \text{and} \quad k = \sum_{j=1}^{n_i} k_j \quad (9)$$

In this expression $p(n_j, k_j)$ is the probability of accessing $k_j$ sensitive documents of $T_j$ in $n_j$ searches of database $D_j$.

The capacity of the leakage channel for all targets $T$ is then the vector

$$C^T = [C^{T_1}(n_1, k_1), C^{T_2}(n_2, k_2), \ldots, C^T(n_q, k_q)] \quad (10)$$

Next, consider the case of a single target, we drop the index identifying the target, and examine several cases:

1) All sensitive documents share a $(\text{key}, \text{value})$ pair, e.g., all documents contain the key-value pair assigning a code name to a patient.

2) All documents of interest are linked together by pairs of documents sharing a unique $(\text{key}, \text{value})$ pair. This is the case of documents $R^n_k$ and $R^n_h$ in our example. In this case there is several possibilities:
   a) The attacker determines the $(\text{key}, \text{value})$ pair of the head of the list and follows the chain;
   b) The attacker determines the $(\text{key}, \text{value})$ pair of one of the documents in the set; thus, it can discover either the downstream or the upstream sensitive documents. This is similar to the previous case, once the document containing the $(\text{key}, \text{value})$ pair is found it acts as a partial head of the list for the documents in the chain;
   c) The attacker determines two $(\text{key}, \text{value})$ pairs of one document thus, it can discover both the upstream and downstream sensitive documents.

Now we analyze the capacity of an $n$-leakage channel when the malicious insider has different means to identify sensitive information and different targets, and assume that she examines $N$ documents and only $K$ of them contain sensitive information.

A. All sensitive documents share a $(\text{key}, \text{value})$ pair. Assuming that the attacker maintains a list of the databases she has already explored, we have a hypergeometric distribution of successes. The problem is sampling without replacement and the probability of finding $k$ sensitive documents in $n$ DB searches, $p(n, k)$ is

$$p(n, k) = \frac{{K \choose k} \times (N-K \choose n-k)}{{N \choose n}} \quad (11)$$

The mean value and the variance of $p(n,k)$ are

$$\mu_p = n \frac{K}{N} \quad \text{and} \quad \sigma_p = \sqrt{n \frac{K}{N} \times \frac{N-K}{N} \times \frac{N-n}{N-1}} \quad (12)$$

The capacity of an $n$-leakage channel $C_{\text{A}}(n, k)$ in this case is given by Equation 11

$$C_{\text{A}}(n, k) = \frac{{K \choose k} \times (N-K \choose n-k)}{{N \choose n}} \quad (13)$$

B. Chained documents. The attacker locates first the head of the list including the $(\text{keyhead}, \text{valuehead})$ pair, then follows the chain by searching for documents containing the pairs $(\text{keyhead}, \text{valuehead})$ and $(\text{keynext}, \text{valuenext})$, until
she identifies the document containing the next two pairs and so on.

To identify the document containing \( \langle \text{key}\text{head}, \text{value}\text{head} \rangle \) the attacker must search \( N \) documents and only one contains the desired \( \langle \text{key}, \text{value} \rangle \) pair. Call \( p_h(n_0) \) the probability of locating the head of the list in \( n_0 \) trials. In this case \( K = 1 \) and \( k = 1 \) and according to Equation 11 \( p_h(n_0) \) is

\[
p_h(n_0, 1) = \frac{(N-1)}{(N)} = \frac{n_0}{N(N-n_0+1)}. \tag{14}
\]

Then the probability of locating a second document of the list in \( n_1 \) trials, \( p_{h_1}(n_1, 1) \) is

\[
p_{h_1}(n_1, 1) = \frac{(N-2)}{(N)} = \frac{n_1}{N(N-n_1+1)}. \tag{15}
\]

It follows that \( p_s(n_0, n_1, \ldots, n_{s-1}) \), the probability of locating \( s \) consecutive sensitive documents in \( n_0, n_1, \ldots, n_{s-1} \) trials is

\[
p_s(n_0, n_1, \ldots, n_{s-1}) = \prod_{j=0}^{s-1} \frac{(N-j)}{(N)} = \prod_{j=0}^{s-1} \frac{n_j}{N(N-n_j+1)}. \tag{16}
\]

It is plausible that the attacker follows a wrong chain; for example, the pair \( \langle \text{Age}, 35 \rangle \) may be present in multiple documents and point the attacker to documents pertinent to a target other than the desired one. Thus, in this case we can only provide an upper bound for the capacity of an \( n \)-leakage channel \( C_B(n, q) \)

\[
C_B(n, s) \leq \prod_{j=0}^{s-1} \frac{n_j}{N(N-n_j+1)} \text{ with } N = \sum_{j=0}^{s-1} n_j. \tag{17}
\]

C. Chained documents. The attacker identifies two \( \langle \text{key}, \text{value} \rangle \) pairs, one pointing upstream \( \langle \text{keyups}, \text{valueups} \rangle \) and one downstream \( \langle \text{keydowns}, \text{valuedowns} \rangle \). First, a search for the document containing both \( \langle \text{key}, \text{value} \rangle \) pairs is conducted. The probability of locating this document in \( n_0 \) trials is equal to \( p_h(n_0, 1) \) given by Equation 14. Then the search is conducted for documents containing either \( \langle \text{keyups}, \text{valueups} \rangle \) or \( \langle \text{keydowns}, \text{valuedowns} \rangle \). It is plausible that the attacker follows a wrong chain in either search thus,

\[
C_C(n, s) \leq C_B(n, s). \tag{18}
\]

3.2 Timing Analysis

The attacker is not only constrained by the number of trials, but also by the time necessary to achieve his objectives. One option for the attacker is to have a script and carry out database searches in parallel to reduce the exposure time. If \( X_1, X_2, \ldots, X_N \) are random variables and \( X_i \) represents the search time for database \( D_i \), then \( T_N \), the time for searching in parallel the \( n \) databases \( D_1, D_2, \ldots, D_N \), is

\[
T_N = \max(X_1, X_2, \ldots, X_N). \tag{19}
\]

When \( X_1, X_2, \ldots, X_N \) are independent random variables and have a common distribution function \( F_X(t) \) then the distribution function of \( T_N \) is given by \[21\]

\[
F_{T_N}(t) = \left[F_X(t) \right]^N. \tag{20}
\]

The expected value of \( T_N \) is

\[
\bar{T} = E[T_N] = N \int_0^{\infty} t F_X^{-1}(t) dF_X(t). \tag{21}
\]

Uniform distribution of the search time. When little is known about a random variable \( X \), except its range \( [a, b] \), a uniform distribution is used to model \( X \),

\[
F_X(t) = \Pr[X \leq t] = \begin{cases} 0 & t < a, \\ \frac{t-a}{b-a} & a \leq t \leq b, \\ 1 & t > b. \end{cases} \tag{22}
\]

According to Equation 21, the time to search in parallel the \( N \) databases when the searching times are uniformly distributed is

\[
T_{\text{uniform}} = N \int_0^{\infty} t (t-a)^{N-1} \frac{1}{b-a} dt = b \frac{b-a}{(N+1)}. \tag{23}
\]

Normal distribution of the search time. Searching a database requires the comparison of the search pattern with multiple documents thus, the search time is the sum of a large number of individual operations. By virtue of the central limit theorem, the distribution of a random variable \( X \) which is the sum of a large number of quantities is normal. The probability density function of a normal distribution with mean \( \mu \) and variance \( \sigma \) is

\[
f_X(t) = \frac{1}{\sigma \sqrt{2\pi}} \exp\left(-\frac{(t-\mu)^2}{2\sigma^2}\right). \tag{24}
\]

There is no closed form for the normal distribution function except for the case of the standard normal distribution when \( \mu = 0 \) and \( \sigma = 1 \). In this case

\[
F_X(t) = \frac{1}{\sqrt{2\pi}} \int_0^t \exp(-\frac{x^2}{2}) dx. \tag{25}
\]

We use a result from [22] to compute the average time

\[
\bar{T}_{\text{stdnormal}} = (2 \log N)^{\frac{1}{2}} - \frac{1}{2} (2 \log N)^{-\frac{1}{2}} \times (\log \log N + \log 4\pi - 2C) + O([\log(N)]^{-1}) \tag{26}
\]

with \( C = 0.577 \) the Euler’s constant.

The obvious method for limiting the capacity of a leakage channel is to trigger alarms when the amount of resources used in a given period by an employee or by someone with access to the credentials of an employee reaches a threshold enforced by the trusted security base of the system. A statistical analysis based on approximate query processing discussed in Section 5 could help the cloud service provider to determine the probabilities of finding sensitive documents in different collections and estimate the number of trials and the time required by an attacker to gather sensitive information close to the capacity of the leakage channels.
4 USING DISINFORMATION TO LIMIT THE CAPACITY OF LEAKAGE CHANNELS

Disinformation in the context of NoSQL databases means document replication combined with the alteration of sensitive \( (key, value) \) pairs to limit the ability of an attacker to identify the true value for a given key. The use of disinformation [26] is a last resort method for limiting sensitive information leakage. Indeed, this solution requires changes of the original database that can be only done by the database owner prior to uploading the data to the cloud. The method also requires a trusted proxy to filter out the fictitious data in the answer to the query.

The indiscriminate replication of all collection documents increases the storage space dramatically as well as the response time for aggregate queries by a factor at least equal to the replication index. The replication index is the cardinality of the set of documents created to hide the sensitive information in an original document. The larger the replication index, the more difficult it is for an attacker to identify the sensitive information, but the larger the storage for the expanded collection.

For example, a 100 TB collection becomes 1 PB collection when the replication index of every document in the collection is equal to ten; the query response time increases in average by an order of magnitude. It follows that replication must be selective, it should only cover documents with sensitive information and could only be applied for relatively small databases.

Disinformation reduces the capacity of leakage channels; for example, a replication factor of ten reduces the capacity of the leakage channel in Equation 11 by an order of magnitude but it also increases the query processing time with an amount related to the effort of identifying the disinformation documents in the response to the query.

A secure proxy like the one in [16] mediates the interaction between clients and the DBaaS server. The proxy intercepts the client queries, transfers them to the encrypted queries and passes them to the cloud DBaaS server which responds with a combination of valid and forged documents. The proxy decrypts the query response and filters out the disinformation documents and forwards the desired document to the user’s application.

This method is not only useful to identify disinformation documents but also to conduct integrity verification using tamper-resistant algorithms. A Message Authentication Code (MAC), also known as a tag confirms that a message comes from the stated sender thus, is authentic and has not been altered. Figure 1. shows this process carried out by the owner of the data:

1. Hash functions are applied to the original and disinformation documents.
2. A new attribute \( \langle e\text{Tag}, E_k(d_i) \rangle \) is appended to each document \( d_i \).
3. The tag is encrypted. Only an authorized database user can decrypt the tag and identify the original document.

Fig. 2: The hashing rate of four popular cryptographic hash functions expressed in KB/second for different document sizes.

The algorithm can use one of several crypto-hash functions including MD5, SHA1, SHA256, and RIPEMD. The time to compute the hash value is a function of the document size. Figure 2 shows the performance of the four popular cryptographic hash functions for a range of document sizes. Based on these results we concluded that SHA1 is the best option.

5 SADS - SENSITIVITY ANALYSIS BASED ON DATA SAMPLING

The documents in the NoSQL databases of a data warehouse include items with different degrees of intrinsic sensitivity and different domains. The intrinsic information quantifies the danger posed by the indiscriminate disclosure of information. The domain quantifies the likelihood that a \( \langle \text{attribute}, \text{value} \rangle \) pair is present in multiple databases.

A large domain increases the capacity of the information leakage channel discussed in Section 3. For example, records containing the Social Security Number (SSN) are likely to be present in health, financial, personnel records, as well as, records maintained by credit scoring agencies, motor vehicle and passport services, airlines and many other organizations with information about an individual.

The goal of sensitivity analysis is to determine the level of vulnerability resulting from correlation of sensitive information in various databases of a public cloud data warehouse and to support measures for limiting the information leakage. Given the massive amounts of data maintained by a public cloud a brute force approach to sensitivity analysis is a hopelessly daunting tasks.

SADS, the solution we proposed is based on Approximate Query Processing (AQP), a technique used by On-Line Analytical Processing applications to extract information
from massive datasets. The response time to a query can be prohibitive thus, limiting the usefulness of data analytics. Many such applications are latency sensitive and in some cases, e.g., in case of exploratory investigations, it is preferable to have sooner an approximate answer to a query than an accurate answer later. In such cases the sampling [15] offers a tempting alternative and, as shown in this section, can also be useful for limiting the information leakage.

**SADS.** Sensitivity analysis has several stages: (i) establish sensitivity levels; (ii) establish the domain of different keys related to sensitive information; (iii) determine the number of collection documents at each sensitivity level. The last two stages of the sensitivity analysis require an examination of all collection documents, a rather slow process. To facilitate fast sensitivity analysis, we shall use samples of the collection and report the estimation errors.

An initial step of sensitivity analysis of the documents in one database can be carried out by the database owner and classify the documents in several classes based on the intrinsic information they could leak. The results of this analysis can be used in several ways to limit the information leakage:

1. Identify data items that can be encrypted to limit the ability of an insider to correlate sensitive information in multiple documents. For example, encrypt records which encode the randomly selected PatientID in case of health records.
2. Identify and rename the key in <key, value> pairs to prevent correlations. For example, instead of “SSN” (Social Security Number) use “PIC” (Personal Identification Code).
3. Selectively apply disinformation to the collection documents.
4. Identify sensitive information and flag repeated queries that search for sensitive information.

A cloud service provider can offer a comprehensive sensitivity analysis service. This service will assess the domain of the key in <key, value> pairs in all databases hosted by the system. It should be stressed that there is no full proof method to completely eliminate the information leakage, the sensitivity analysis can only limit it.

**AQP and Sampling.** AQP is based on sampling techniques for providing approximate responses to aggregated queries alongside estimations of the implicit error produced by this method. An aggregate query calls aggregate functions to return a meaningful computed summary of specific attributes of a group of documents. Common aggregate functions are: Average, Max, Min, and Count [23].

An AQP system supplies confidence intervals indicating the uncertainty of approximate answers. Confidence Intervals (CI) represent the range of values centered at a known sample mean and used to calculate error bounds. Indeed, an approximate answer without the specification of the errors due to sampling rather than full database search would not be useful.

Sampling can be done with or without replacement. In sampling without replacement (disjoint samples), any two samples are independent whereas in sampling with replacement, sample values are dependent. The sample size should be increased when the estimation error of the AQP method is higher than an acceptable threshold during the sampling phase. The results produced by resampling without replacement from the larger sample set are dependent on the original sample set. The resampling process can be repeated to balance estimation errors while limiting the processing time required by the bootstrapping method [15], [29].

Collection samples consist of randomly selected documents from the original collection. Queries can be conducted in parallel on such samples. Given $C$, the set of documents in the collection and $S$, the set of documents in a sample used by the AQP method, the scaling factor, $\sigma$, is defined as:

$$\sigma = \frac{|C|}{|S|}$$

The smaller the sample size, the larger is $\sigma$, and the shorter is the response time to a query posed to the sample, but also the larger are the estimation errors based on this sample.

Let $S$ be a set of $n$ sensitivity classes of documents in $C$, $S = \{s_1, s_2, \ldots, s_n\}$. Call $c_i$ the count of documents classified in sensitivity class $s_i$ with $|C| = \sum_{i=1}^{n} c_i$. Given the aggregate query $\theta$, let $\hat{\theta}$ be the corresponding approximate query carried out using the documents in sample $S$.

The response to the approximate query $\hat{\theta}$ may only include documents in $m \leq n$ sensitivity classes $\hat{s}_i$ of the set $\hat{S} = \{\hat{s}_1, \hat{s}_2, \ldots, \hat{s}_n\}$. Call $\hat{c}_i \leq c_i$ the count of documents classified in sensitivity class $\hat{s}_i$. Then $|\hat{S}| = \sum_{i=1}^{m} \hat{c}_i$.

**Sampling errors.** A key element of any AQP system is to provide error bounds for the approximative results, allowing the user to decide whether the results are acceptable. The Sampling-based Approximate Query Processing (S-AQP) with guaranteed accuracy provides bounds on the error caused by sampling [15].

The distribution converges into the standard normal random distribution $N(0, 1)$ as $n$, the number of elements in the sample, goes to infinity. The sampling error for sensitivity class $s_i$ is

$$\hat{e}_i = 100 \frac{c_i - \hat{c}_i}{c_i}. \quad (28)$$

The error vector due to sampling is

$$E = (\hat{e}_1, \hat{e}_2, \ldots, \hat{e}_n). \quad (29)$$

In a uniform random sampling $n-m$ classes may not appear in the response to the query and components of the error vector for missing classes are 100%.

**Fig. 3:** Bound tightness comparison obtained by using Markov, Chebyshev inequalities and close-form CLT.

We use a close-form Central Limit Theorem (CLT) and Markov and Chebyshev inequalities to get the tightest bounds. The tightness of the bounds resulted from the
three aforementioned approaches is illustrated in Figure 3. Markov chain provides larger deviation bounds than Chebyshev inequality. Close-form CLT provides the tightest bound among these three approaches [35].

**Experimental results.** The results reported in this section are restricted to sensitivity analysis based solely on intrinsic information and we report on two groups of experiments to: (a) investigate the effect of sample size on the estimation errors and (b) study the effectiveness of the selective disinformation.

These experiments were conducted on a cluster of 100 AWS EC2 instances (t2.large) with two vCPU, 8 GB memory, and the Linux kernel version 4.4.0-59-generic. MongoDB version 3.2.7 was used as the NoSQL server. MongoDB supports a variety of storage engines designed and optimized for various workloads. The storage engine is responsible for data storage both in memory and on disk; we chose WiredTiger storage engine. The OPE and AHOM cryptosystems are implemented locally and other crypto modules are imported from OpenSSL version 1.0.2g.

The relationship between sample size and estimation error. We created four sets of random samples from an original collection of $10^7$ documents. Each one of the four sets included 100 random samples with $10^2$, $10^3$, $10^4$, and $10^5$ documents, respectively. The samples were selected with and without replacement. Figure 4 displays the error for different sample size for the two different sampling modes.

The measurement results show that samples without replacement exhibit slightly more accurate results than samples with replacement. For instance, the average error percentage is 0.22% for the largest sample of $10^5$ documents, whereas the error is 5.08% for the smallest sample size of 100 documents. We concluded that a scaling factor of $10^3$ is perfectly suitable. This scaling factor is likely to reduce the average query response time by two orders of magnitude.

The effectiveness of selective disinformation. The experiments have three objectives: (a) Investigate the accuracy of sensitivity analysis for collections with multiple classes of documents; (b) Study the effect of sampling on query response time; and (c) Assess the effectiveness of the selective disinformation.

In these experiments, we created a collection of ten million documents in eight sensitivity classes. Table 1 shows the eight sensitivity classes and the count and percentage of documents in each sensitivity class.

<table>
<thead>
<tr>
<th>Class ($c_i$)</th>
<th>Cardinality($c_i$)</th>
<th>Percentage</th>
<th>Difference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Top Secret</td>
<td>782 471</td>
<td>7.823</td>
<td>-2.32</td>
</tr>
<tr>
<td>Secret</td>
<td>1 475 118</td>
<td>14.751</td>
<td>2.16</td>
</tr>
<tr>
<td>Information</td>
<td>3 134 844</td>
<td>31.348</td>
<td>3.66</td>
</tr>
<tr>
<td>Official</td>
<td>1 475 603</td>
<td>14.756</td>
<td>2.16</td>
</tr>
<tr>
<td>Unclassified</td>
<td>783 443</td>
<td>7.834</td>
<td>-3.22</td>
</tr>
<tr>
<td>Clearance</td>
<td>783 024</td>
<td>7.830</td>
<td>-3.22</td>
</tr>
<tr>
<td>Confidential</td>
<td>782 698</td>
<td>7.826</td>
<td>-3.22</td>
</tr>
<tr>
<td>Restricted</td>
<td>782 799</td>
<td>7.828</td>
<td>-3.22</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td>9 945 260</td>
<td>100.00%</td>
<td></td>
</tr>
</tbody>
</table>

Then we sampled the original collection and investigated the accuracy of document classification in the eight sensitivity classes. Table 2, displays the estimated cardinality of each class, when the sample size is $10^4$. In this case the scaling factor is

$$\sigma = \frac{10^7}{10^4} = 10^3$$ (30)

and the overall error is less than 0.55% as only less than 55,000 out of 107 documents have been misclassified. We expected a query process speedup of the same order as $\sigma$ and, it is indeed 1300 as we shall see next.

Once the error was estimated the experiment continued with the estimation of the query response time, defined as the interval between the time when the server receives a query and the time it starts forwarding the result. Most database servers cache the most recently used data to reduce the response time latency. In our experiments query caching and prefetching was disabled to force the query optimizer to serve the next matching queries directly from the database.

The aggregate query displayed in Figure 5 is used for computation of cardinality and percentage of each sensitivity class.

The results show that the average speedup due sampling is better than linear and the estimation errors are quite low. Both metrics are plotted for the four sample size in Figure 6. For the largest sample of $10^5$ documents, the average processing time is 93 ms.
collection (including $10^7$ documents) the processing time of the same query is $14000$ ms; the speedup is $150$.

Lastly, we used the sensitivity analysis to generate disinformation. A disinformation replication factor $V$ is assigned to each class according to sensitivity. For example, knowing the approximated cardinality value $c_i$ of the number of documents in each sample class we choose replication factors of $100, 25, 0, 5, 10, 0, 50$ and $15$ for “Top Secret”, “Secret”, “Information”, “Official”, “Unclassified”, “Clearance”, “Confidential” and “Restricted” classes, respectively, from Table 2. The expansion factor $E$ is defined as the ratio of the cardinality of the collection including disinformation documents to that of the original collection. In this example, the expansion factor is $E = 0.18$, while indiscriminate disinformation insertion leads to an expansion factor $E = 100$.

The overhead is drastically reduced while providing a similar leakage prevention level.

The conclusion of our experiments is that SADS is a powerful method to substantially reduce query latency with bounded and small estimation errors. SADS with uniform random sampling provides sensible results for classification aggregate query workload, with a sensible compromise between sample size and query latency. However, for queries with different workloads such as aggregate functions that involve multiple correlated collections, the uniform sampling cannot provide accurate responses and we designed a new technique for biased sampling solution for this problem. In the next section, we highlight approximated answers for correlated collections.

### 6 Warehouse Information Leakage

Investigation of potential leakage in a data warehouse requires cross-correlations among all datasets, a truly daunting task due to colossal amount of data and the huge computational effort. A warehouse hosting $n$ databases, each database with $m$ collections, and each collection containing $q$ documents requires $N_c = (m \times n \times q)^2$ correlations. For example, when $m = 10^3$, $n = 10^3$, and $q = 10^9$ then $N_c = 10^{16}$. Can the sampling methods discussed in Section 5 be extended to a cloud data warehouse hosting a large number of databases? This is the topic explored in this section.

Ultimately, we wish to understand the relationships among the collections in a data warehouse and among the attributes of the documents. These relationships can be captured by two networks: (1) the collection network $\Gamma_C$; and (2) the attribute network $\Gamma_A$. Construction of these two networks follows the same ideas discussed earlier, sampling the databases followed by correlations among samples and the determination of the estimation errors.

The vertices and the links of $\Gamma_C$, the collection network, are the collections and the identifier attributes, respectively. Two vertices $i$ and $j$ are connected if they share a number of identifier attributes. The link connecting the two vertices is labeled with the list of common attributes of the two collections. In our example the number of vertices is $m$.

The vertices and the links of $\Gamma_A$, the attribute network, are the attributes and the list of documents sharing attributes, respectively. Two vertices $i$ and $j$ are connected through a link if they appear together in one or more documents. The link between the two vertices is labeled with the Id of the documents containing the common identifier attributes. An identifier attribute could be postal address, phone number, social security number, patient ID, and so on for a collection related to healthcare, utilities, financial records, etc.

We suspect that $\Gamma_C$ and $\Gamma_A$ are scale-free networks with a power-law distribution of node degrees [37]. Networks with a power-law distribution of node degrees appear naturally in social networks and other virtual organizations. Such organizations are inherently heterogeneous, there are a few highly connected entities and a very large number of entities with a few connections. Several instances of virtual organizations, as well as man-made systems, seem to enjoy this type of organization.

![Fig. 5](image-url) The aggregate query for sensitivity analysis of a given collection including the original or sample datasets.

![Fig. 6](image-url) The performance of SADS classification over 100 sample sets with $10^2, 10^4, 10^6$ and $10^8$ documents: (a) the processing time of aggregate query; (b) achieved speedup.
In a scale-free organization the probability \( p(k) \) that an entity interacts with \( k \) other entities decays as a power law

\[
p(k) \approx k^{-\gamma},
\]

with \( \gamma \) a constant and \( k \) a positive integer. We only consider the discrete case when the probability density function is

\[
p(k) = a f(k) = k^{-\gamma} \quad \text{and the constant} \quad a = 1/\zeta(\gamma, k_{\text{min}}) \quad \text{thus,}
\]

\[
p(k) = \frac{1}{\zeta(\gamma, k_{\text{min}})} k^{-\gamma}.
\]  

In this expression \( k_{\text{min}} \) is the lowest degree of any node, and in our discussion we assume that \( k_{\text{min}} = 1 \); \( \zeta \) is the Hurvitz zeta function\(^1\).

The degree distribution of scale-free networks follows a power law

\[
\zeta(\gamma, k_{\text{min}}) = \sum_{n=0}^{\infty} \frac{1}{(k_{\text{min}} + n)^\gamma} = \sum_{n=0}^{\infty} \frac{1}{(1 + n)^\gamma}.
\]  

Figure 7 shows the graph of a scale-free network. The average distance \( d \) between the \( P \) nodes of a scale-free network, also referred to as the diameter of the scale-free network, scales as \( \ln P \).

Empirical data confirm the existence of scale-free organization in many instances where multiple entities are interconnected with one another. For example, the power grid of the Western US has some 5,000 nodes representing power generating stations; in this scale-free network \( \gamma \approx 4 \). The collaborative graph of movie actors where links are present if two actors were ever cast in the same movie follows the power law with \( \gamma \approx 2.3 \)[38]. Recent studies indicate that \( \gamma \approx 3 \) for the citation of scientific papers [39].

**Correlations among collections.** We consider a 2-way correlation reflecting an unintentional join relation of collections \( C_L \) and \( C_R \). This relation is a result of a common identifier attribute, namely linkage attribute. A k-way correlation with \( k \) \( > \) 2 is a k-way join relation between a sequence of \( k \) correlated collections \( C_1, C_2, \ldots, C_k \). A k-way correlation is equivalent to a combination of multiple 2-way correlations [28], [36].

**Biased sampling.** A new sampling strategy is discussed next as uniform random sampling leads to large errors. Biased sampling takes into account the frequency of values [40] in the original dataset. Biased sampling creates sample sets with respect to the repetition frequency of each value of the intended attribute. The higher the frequency of occurrence, the higher the probability to be included in the sample set. Furthermore, infrequent collection documents have a small contribution to the sample set. However, if there are large numbers of infrequent values their impact adds up.

We customized the biased sampling algorithm consistent with cross-correlation analysis queries which are join-driven to probe and extract the leaked attributes from the given collections. For any 2-way join query there are two collections: the left and the right collection. The cross-correlation analytical query returns a new set of attributes from the right collection based on the evaluation of the join-predicate.

Tunable thresholds \( T \) for collection \( C \) can balance the sample size and the accuracy. Documents occurring with frequency \( f_v > T \) are added to the sample, while those with frequency \( f_v \leq T \) are included with probability \( p_v = f_v/T \). Higher values of \( T \) result in a smaller sample set.

Sometimes we wish to bias the sampling by adjusting the thresholds of two collections as seen in Equation 34. This equation shows \( c_v \) the cross-correlation using biased sampling for an attribute value \( v \) of two collections \( C_L \) and \( C_R \) with the threshold parameters \( T_L \) and \( T_R \), respectively.

\[
\begin{align*}
\text{if } f_L(v) \geq T_L \text{ and } f_R(v) \geq T_R & : c_v = f_L(v).f_R(v) \\
\text{if } f_L(v) < T_L \text{ and } f_R(v) \geq T_R & : c_v = T_L.f_R(v) \\
\text{if } f_L(v) \geq T_L \text{ and } f_R(v) < T_R & : c_v = f_L(v).R(v).\max(T_L/T_L(v), T_R/T_R(v)) \\
\text{if } f_L(v) < T_L \text{ and } f_R(v) < T_R & : c_v = 0
\end{align*}
\]

We adjust the threshold parameters of the biased sampling algorithm to generate a larger sample set from the right collection. \( T_R \) is adjusted to be significantly smaller than \( T_L \) to increase selection probability thus, a larger sample size from \( C_R \). With this adjustment the sizes of the two samples are different. The cross-correlation analysis uses the samples created by this process.

An instance of correlation extractor query is displayed in Figure 8. After samples are created, the 2-way cross-correlation analysis query is processed over the samples instead of the original data.

**Fig. 7:** A scale-free network is non-homogeneous; the majority of vertices of a graph model of a scale-free network have a low degree and only a few vertices are connected to a large number of edges; the majority of the vertices are directly connected with the vertices with the highest degree.

**Fig. 8:** The aggregation join query for discovery of attributes from the right collection based on evaluation of equality check on the value of the common attribute.

1. The Hurvitz zeta function \( \zeta(s,q) = \sum_{n=0}^{\infty} \frac{1}{(q + n)^s} \) for \( s, q \in \mathbb{C} \) and \( \Re(s) > 1 \) and \( \Re(q) > 0 \). The Riemann zeta function is \( \zeta(s,1) \).
respectively. The exact correlation between $C_L, C_R$ is denoted by $C_{LR}$ while $S_{LR}$ is the approximated value computed for $S_L$ and $S_R$. Using biased sampling, the correlation size approximation is computed by $\hat{C} = \sum c_v$. The scaling factor $\frac{1}{\min(p_x, p_y)}$ is used to scale up the result from the size of the samples to the size of the original dataset.

**Experimental results.** We use four correlated databases from social media, phone directory, medical, and financial areas. Each collection includes $10^7$ documents. The known pairwise correlation is shown in Figure 9.

The proposed estimation method based on biased sampling is evaluated on different datasets and compared with the random sample selection. The optimized biased sampling provides more accurate results and reduces the processing time. The frequency of concurrence in each dataset can be computed offline.

![Social network Profile](Image)

![Health](Image)

![Finance](Image)

![Phone directory](Image)

Fig. 9: The estimation algorithm uses four datasets each with $10^7$ documents and with a pre-defined level of correlations: 50.34% between social network profile and the phone directory; 31.18% between health and financial collections; and 41.7% between health collection and the phone directory.

![Size of cross-correlation](Image)

Fig. 10: The approximation of the cross-correlation size using random and biased sampling.

The comparison between random sampling and the proposed method in Figure 10 shows that biased sampling performs better than random sampling. The estimation errors for the biased sampling method are very low, almost zero for two of the three database correlations.

The exact cross-correlation between social media and phone directory collections requires almost 12 000 seconds while the time is reduced to 1.5 seconds with 1% error by this sampling method.

7 **Conclusions and future work**

The impact of information leakage will most likely grow in the future not only for public clouds, but also for private clouds. Indeed, more organizations and government agencies transition to private and hybrid clouds with the belief that a cloud could offer enhanced security and prevent data theft. The sad reality is that there are no full proof methods to prevent information leakage as cloud data encryption discussed in Section 2 has serious limitations.

The sensitivity analysis based on data sampling, inspired by approximate query processing, introduced in Section 5 offers a glimmer of hope. The sensitivity analysis identifies the most valuable information to be protected and offers some guidance on how to protect against insider attacks.

Attribute correlation among the databases of a cloud warehouse, involves processing enormous amounts of data and brute force methods are hopeless. The method based on heterogeneous biased data sampling has a reasonable level of accuracy. The optimum sample size results in substantial speedup with results close to the exact value.

We suggest introduction of leakage detection cloud services that can offer guidance to organizations on how to better protect their data and minimize the risks of information leakage. Sensitivity and cross-correlation analysis at cloud warehouse level can only be conducted by a CSP with access to all datasets. The extension of individual Service Level Agreements to include a clause related to information leakage protection will allow CSPs to periodically compute the correlations necessary to construct the two networks discussed in Section 6.

**Future work.** The authors are further developing an information theoretic framework for the analysis of information leakage based on the capacity of a leakage channel introduced in Section 3. We are also developing algorithms to construct the collection and the attribute networks and will investigate if our assumption that $\Gamma_C$ and $\Gamma_A$ are indeed scale-free networks is valid and will continue to experiment with large collections of documents.

**Acknowledgments**

The authors wish to express their gratitude to anonymous reviewers whose comments and suggestions contributed to significant improvements of the paper.

**References**


Mohammad Ahmadian Got his Ph.D. in 2017 from the Computer Science Department at University of Central Florida. His research interests are in the area of public cloud security.

Dan C. Marinescu During the period 1984-2001 Dan Marinescu was an Associate and then Full Professor in the Computer Science Department at Purdue University in West Lafayette, Indiana. Since August 2001 he is a Professor of Computer Science at University of Central Florida. He is conducting research in parallel and distributed systems, complex systems, and quantum information processing. Dan Marinescu has published more than 220 papers in journals and refereed conference proceedings. He has also published several books including: “Internet-based Workflow Management” published by Willy-Interscience in 2002, “Cloud Computing; Theory and Practice” published by Morgan Kaufmann in 2013, “Complex Systems and Clouds; A Self-organization and Self-management Perspective” published by Morgan Kaufmann in 2016 and “Cloud Computing; Theory and Practice” Second Edition, published by Morgan Kaufmann in November 2017.