A Robust Approach for Data Leakage Assessment*

[1]H Abdel-Fatao, [1]V. M. Nofong, [1]L. K. Agbeyeye, [1]Y. M. Umaru, and [1]B. K. Alese
[1]University of Mines and Technology (UMaT), Tarkwa Ghana

Abdel-Fatao, H., Nofong, M.V., Ledi, K. A., Umaru, M. Y., and Alese B. K. (2021), "A Robust Approach for Data Leakage Assessment", *Ghana Journal of Technology*, Vol. 5, No. 2, pp. 48 - 59.

Abstract

Modern market practices encourage organisations to outsource major data management responsibilities of key data-centric processes to third-party service providers (agents). A major challenge that confronts organisations that embrace this philosophy is how to detect leakage of outsourced data and trace its source. To address the challenge, this study proposes a robust approach for establishing the provenance of organisational data leakage. Specifically, a three-stage approach is developed. Firstly, the dataset to be outsourced is transformed to make it unique. The uniqueness of the dataset is achieved by subjecting it to a composite of three techniques namely smart distribution, the addition of fake objects and application of transposition encoding. Secondly, when leakage occurs, an effective *how* and *where* data lineage technique is used by a detection model to establish the occurrence of leakage. Finally, the leakage source is traced using the guilt assessment model which consists of the probabilistic, transposition key and fake object tests. The proposed approach is evaluated using real-world credit card datasets. The results obtained from the leakage scenarios showed 100% confidence in detecting leakages and identifying the culpable agent of leakage.

Keywords: Data Leakage, Data Allocation, Agent Guilt Assessment, Relational Database

1 Introduction

Recent decades have witnessed the accumulation of massive amounts of datasets from various sources owing to the advent of Web 2.0, now 3.0; and the development of reliable global telecommunication infrastructure. Thanks to this technological shift, analysis of large datasets is now at the forefront of many evolutionary breakthroughs in numerous fields today. The vast majority of modern-day organisations in multitude of domains such as healthcare, public administration, business, academic and industrial research, rely on some form of data analysis for important strategic decision making (Kowalczyk and Buxman, 2014; Ram *et. al.*, 2016). The influx of data collection and analytics tools in businesses nowadays has transformed the way business is conducted to produce improved income streams. Data-centric businesses are able to collect better market and customer intelligence, enhance customer experience as well as improve internal efficiency and operations (Kowalczyk and Buxman, 2014; Ram *et. al.*, 2016).

A major challenge that plagues most data-centric organisations is how to store, manage and conduct, in a timely manner, useful analysis on huge data streams that they continue to generate (Khan *et. Al.*, 2014, Mazumdar *et. Al.*, 2019). In fact, many of these organisations lack adequate storage capacity to contain and analyse data generated in-house. To address this challenge, modern market practices encourage business owners to adopt the strategy Business Process Outsourcing (BPO). BPO consists in subcontracting management responsibilities of key data-driven business processes of an organisation to specialist third-party service providers (agents). These third-party service providers involved in BPO constitute agencies that render services such as marketing, advertising, human resource, research, etc. for organisations. BPO is vital for survival of many organisations in today's extremely competitive business environment as it relates to the task of efficient organisational designs, in terms of cost reduction, productivity, growth and innovative capabilities (Chanvarasuth, 2008; Espino-Rodríguez and Ramírez-Fierro, 2017).

BPO essentially requires the agents to be granted access to participating organisations' sensitive information to carry out requested services. For instance, a business entity may be required to issue customer records to an agency in order to conduct customer behavioural analysis; a hospital may need to deliver patient records to a research institution towards the development of treatments for diseases etc. Data outsourcing for BPO is therefore deeply entrenched in the establishment of trust among participating parties (Greenberg *et. al.*, 2008). In most cases, legally binding service-level agreements are documented to ensure that agents meet owners' expectations, as well as guarantee confidentiality and sanctity of outsourced data.

As more organisations embrace the BPO ideology, the volume of sensitive data being shared with and outsourced to agents during BPOs continues to grow exponentially. These sensitive data may include datasets containing records on intellectual property, financial information, patient health record, personal credit-card data, among others (Shabtai *et al., 2012*). Despite the numerous proven benefits of BPO and the stringent security measures that are taken to ensure its integrity, service providers sometimes

*Manuscript received July 1, 2020
Revised version accepted January 17, 2021

intentionally or inadvertently distribute confidential company information to unauthorized hands; a phenomenon referred to as *data leakage*.

Data leakage can be defined as unauthorized, intentional (malicious) or inadvertent appropriation and transmission of classified data (information) from a computer or data centre within an organisation to an external entity. Data leakage can occur electronically or physically and often accomplished by simple memorization and recall of information, physical removal of storage media such as disks, tapes and reports, or subtly by data hiding. Data leakage may be initiated by internal (insider leakage) or external (intruder leakage) sources to an organisation (Koti *et. al.*, 2017). The main enabling risk factors include too many users with excessive access privileges (37%), an increasing number of devices with access to sensitive data (36%), and the increasing complexity of information technology (35%), according to Schulze (2018). It may occur as a result of an accidental breach, for instance, when an employee inadvertently transmits confidential information to the wrong recipient of an email. It may also occur when sensitive information gets delivered to adversaries or competitors by disgruntled employees in exchange for huge pay check (corporate espionage).

Data leakage, whether caused by malicious intent or an inadvertent spill by an insider or outsider poses an ongoing challenge for organisational data security. This is because of the many debilitating consequences it can cause to organisations (Solami *et al*., 2020). For instance, data leakage can potentially result in direct losses which involve easily quantifiable tangible damages (Shabtai *et al.*, 2012). These damages may arise from violations of regulations (such as those protecting customer privacy) resulting in fines, settlements; litigation involving crippling lawsuits and massive financial penalties; loss of future sales and plummeting revenues; costs of investigation and remedial restoration fees. Data leakage can also lead to indirect losses which are much harder to quantify and have much broader impact in terms of cost and time such as reduced share prices as a result of negative publicity; damage to a company's goodwill and perpetually tarnished reputation; customer abandonment; and exposure of intellectual property (business plans, code, financial reports, and meeting agendas) to competitors (Shabtai *et al.*, 2012). It is therefore crucial for organisations to implement policies and measures aimed at mitigating its occurrence and to protect themselves against liabilities that might arise as a result of data leakage.

In the light of the ensuing discussion, this study tackles the problem conducting digital forensics in the event of organizational data leakage. More precisely, this paper principally focuses on tackling the problem of establishing data provenance (Buneman *et al.*, 2001; Cheney *et al.*, 2009) in relational databases. Data provenance can generally be defined as the process of detecting and accurately showing that data leakage emanates from a specific agent. Essentially, a data leakage detection and guilt assessment problem are considered herein based on the following scenario adapted from Papadimitriou and Garcia-Molina (2009): *A certain company issues datasets to 3 agents designated as Ag_1, Ag_2 and Ag_3, to conduct various assignments. The company later discovers a suspicious dataset which bears a close similarity to part of the dataset issued to the agents in the possession of a competing organisation. The company then decides to find out whether or not the discovered dataset was in fact its bona fide property, and if so, trace which of the three agents is culpable of leaking the dataset to the competitor.* To tackle this problem of data leakage detection, this study proposes a three-stage sequential approach expounded upon below.

Firstly, the dataset to be handed over to an agent is passed through the transformational model in order to make it unique. The transformational model achieves the desired uniqueness of the dataset by subjecting it to a composite of three techniques namely smart distribution, the addition of fake objects and application of transposition encoding. Secondly, when leakage occurs, an effective *how* and *where* data lineage technique is used by a detection model to establish the occurrence of leakage. Finally, the leakage source is traced using the proposed guilt assessment model which consists of the probabilistic, transposition key and fake object tests. The proposed solution to data leakage detection was evaluated using real-world credit card datasets. The results obtained from the leakage scenarios showed 100% confidence in detecting leakages and identifying the agent responsible for the leakage.

1.1 Related Work

The past two decades have witnessed a devotion of much attention towards research into corporate data leakage detection and guilt assessment by academia and industry, especially in relation to cases involving relational databases. The vast majority of pioneering studies on data leakage detection involved the use of watermarking techniques or digital fingerprinting (Agrawal and Kiernan, 2002; Shabtai *et al.*, 2008; Shabtai *et al.*, 2012; Zhang *et al.*, 2004). Watermarking involves embedding imperceptible information in the form of text, images, audio etc. for right and ownership protection of digital intellectual property (Guo *et al.*, 2006).

Watermarking and/or digital fingerprinting of relational databases are quite proficient for ownership protection, tamper proofing, and proving data integrity. For instance, Agrawal and Kiernan (2002) proposed a watermarking technique that marks only numeric attributes and those that tolerate some changes in their values. The algorithm embeds the watermark bits in the least significant bits (LSB) of the selected attributes of a selected subset of tuples. This technique cannot be used for multi-bit watermarks. The LSB bits in any tuple can be altered without checking data constraints. It is also not resilient to insertion, deletion and alteration attacks.

Sion *et al*. (2003) and Sion *et al*. (2004) proposed a watermarking algorithm that embeds the watermark into the relational database using data portioning technique. The algorithm works in two stages that involve *encoding* and *decoding* of the bits. The algorithm checks the fit tuples determined by the attribute and primary key of a relation. For checking the tuples, a hash function is applied which uses SHA or MD5 algorithm for hashing. Unlike the propositions by Agrawal and Kiernan (2002), the technique proposed by Sion *et al*. (2003) is resilient to alteration and data loss attack. Also, Radu (2004) define some watermark attacking scenarios available for the attacker and suggests design choices to increase watermark safety. However, according to Kamran and Farooq (2018), the investigation by the authors was not detailed.

Zhang *et al.* (2004) proposed a watermarking algorithm in which an identification image is embedded into the relational data for representing the copyright information. This algorithm takes input as the relation R with the attributes as $R(K.A_0, A_1, \ldots, A_n)$ where K is the primary key of the database which is never marked. The pixel values are marked as $I(v_0, v_1, \ldots, v_n)$. The relation R is divided into groups of uniform size equal to the size of the image. The algorithm compares a pixel value with an attribute value of a tuple in a relation. Pixel value (0 to 255) is divided into 3 parts. Three types of watermarks are inserted into the relational data. Else if 0 or 255 are repeatedly present, lots of attribute values will be marked as the same numbers in their decimal. After performing some experiments on images, it was verified that the watermarking algorithm designed by Zhang *et al*. (2004) is resilient to subset selection attack but not resilient to subset alteration, deletion and insertion attacks.

Shehab *et al.* (2008) proposed a watermarking technique of relational databases, which solves the optimization problem based on genetic algorithm and pattern search techniques. The technique is divided into two parts: Watermark encoding and decoding. The watermark encoding is done in three stages namely dataset partitioning, watermark embedding and optimal threshold evaluation. Watermark decoding is also accomplished in three stages namely dataset partitioning, threshold-based decoding and majority voting. For each tuple r in dataset S, the data partitioning algorithm computes a message authenticated code. In single bit encoding, optimization problem is solved by maximizing or minimizing the hiding function, which is based on single bit. If the bit is equal to 0, the problem is considered as a minimization one otherwise, it is considered to be a maximization problem. This algorithm is resilient to various attacks such as deletion, insertion, and alteration at the same time not vulnerable to synchronization errors. It also minimizes the probability of decoding error because of optimal threshold.

Al Solami *et al*., (2020) proposed a fingerprinting technique to data leakage detection in relational databases. In their study, the authors embed hidden signatures inside relational databases, which could either be randomly generated or meaningfully organized sequences of bits. The approach consists of two phases namely the insertion phase and the detection phase. In the insertion phase, a buyer-specific digital signature (fingerprint) is generated for each user of a relational database and embedded in the original data using an embedding algorithm that uses a secret key K. When data leakage occurs, a novel fingerprint decoding algorithm which is resilient against malicious attacks that might try to damage the embedded mark is applied to the suspicious data to detect the embedded fingerprint. Even though the authors showed the robustness of the technique through comparison with some state-of-the-art techniques, their proposed approach still suffer from the short digital watermarking because they are based on similar principles.

All the watermarking and/or digital fingerprinting techniques discussed above involved modification to the original data. These may sometimes affect the outcome of a business process.

To address the aforementioned problem with the watermarking and/or digital fingerprinting techniques, Papadimitriou and Garcia-Molina (2009) proposed a non-perturbation technique. This technique does not require a modification to the original data before being issued to an agent. Instead, data is distributed among the agents in a manner that improves the chance of detecting a leaker during leakage. Two algorithms were proposed in distributing data to agents namely the e-random algorithm and the s-random algorithm. The e-random is used to allocate data during explicit data requests whiles the s-random algorithm is used during the sample data requests. Papadimitriou also introduced the concept of fake objects in which fake but realistic data records are added to the dataset

given out to service providers. The proposed technique is efficient and effective in detecting leakage and identifying the suspect. Nevertheless, the technique assumes that the total number of service providers or agents and their data requested are known in advance before data allocation is performed. This assumption does not hold in real world scenarios.

2 Resources and Methods Used

2.1 Description of Dataset

To tackle the data leakage problem emanating from the scenario described in Section 1, this study assumes that the leaked dataset contains sensitive information such as credit card details. In the light of this assumption, a credit card dataset: *AER Credit Card Data from Book Econometric Analysis*, originally published alongside the 5th edition of William Greene's book Econometric Analysis, is employed to demonstrate the proposed approach for data leakage detection and agent guilt assessment. Fig. 4 typifies a sample of the AER credit card dataset showing six attributes (detailed description of the dataset can be found on the website).

2.2 The Agent Guilt Assessment Model

The agent guilt assessment model developed in this study involves a sequence of three (3) stages namely the *data request model; data transformation model; leakage detection and guilt assessment. E*ach stage of the model is discussed below.

Table 1 Sample AER Credit Card Dataset

CardID	Age	Income	share
UT01143	26.5	3.36	0.002625298
UT01144	43.08333	7.8	0.0066855130000...
UT01145	50.0	4.3	0.03620279
UT01146	35.66667	3.5822	0.1049978
UT01147	43.25	6.72	0.03142812
UT01148	44.5	4.0	0.01795825
UT01149	33.16667	2.3204	0.0005171522
UT01150	30.33333	2.5	0.0356228
UT01151	25.08333	1.5	0.0128
UT01152	31.75	3.5	0.01055514
UT01153	27.16667	4.0	0.0003
UT01154	71.83334	7.7010000000...	0.06251058
UT01155	28.33333	6.6	0.007129848

2.2.1 Data Request Model

The data request model is a functionality that enables agents to demand the type and quantity of data. Data requests fall into two categories namely sample and explicit request (Papadimitriou and Garcia-Molina, 2009; Papadimitriou and Garcia-Molina, 2011). A sample request is made if an agent seeks any subset of records from data space without any predefined condition(s). For instance, a sample request by an agent could be of the form: *Supply records of 20 students for onward analysis*. In contrast, a request is said to be explicit or conditional if the nature of data requested by must fully satisfy all the conditions specified by the agent. For instance, an explicit request by an agent could be of the form: *Supply records of 20 students in the computer science and engineering department for onward analysis.* The data request model implemented in this study handles both sample and conditional requests.

2.2.2 Data Transformation Model

The data transformation model involves three techniques - *data allocation, fake objects addition and transposition encoding*. Fig. 1 is a diagrammatic depiction of the conceptual view of the data transformation model.

Data Allocation

The object of the data allocation technique is to develop a scheme that guarantees that, overlap between datasets issued to agents is minimum. To achieve this, data objects to be allocated to agents are firstly pruned using priority scheduling (Davies and Burns, 2007; Katcher *et al*., 1993) in order to filter data objects returned by the request model. A formal explanation of the technique is given below.

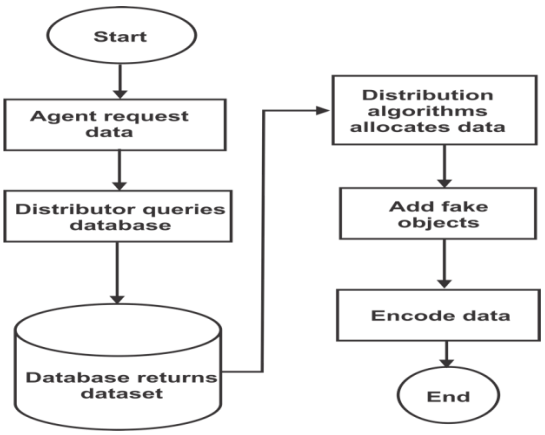

Fig. 1 Conceptual View of the Data Transformation Model

Let $T = \{t_1, t_2, \ldots, t_n\}$ denote a finite set of n data objects in the data distributor's database. Assume an agent denoted by Ag_x, requests data records of size designated as *requestQuantity*, a set of data objects $T(Ag_x) = \{t_1, t_2, \ldots, t_{requestQuantity}\}$ satisfying a set of m conditions $C = \{cond_1, cond_2, \ldots, cond_m\}$ are returned, where $T(Ag_x) \subseteq T$. Algorithm 1 details the steps in the proposed data allocation approach.

Algorithm 1: Data Allocation

Require: $T = \{t_1, t_2, t_3, \ldots, t_n\}$, *requestQuantity* $(C = cond_1, cond_2, \ldots, cond_m)$

Ensure: $T(Ag_x) = \{t_1, t_2, t_3, \ldots, t_{requestQuantity}\}$

1: $T(Ag_x) \leftarrow \emptyset$
2: *temp* $\leftarrow \emptyset$
3: **for** $i = 1, 2, \ldots, n$ **do**
4: **if** t_i satisfies C **then**
5: $temp \cup t_i$
6: **end if**
7: **end for**
8: *temp* $\leftarrow$ *sort(temp)* {sort element in ascending order of allocation frequency}
9: **for** $j = 1, 2, \ldots,$ *requestQuantity* do
10: $t_j = temp[j]$
11: $T(Ag_x) \cup t_j$
12: *update Frequency*(t_j)
13: **end for**
14: **return** $T(Ag_x)$

After initialising the agent set Ag_x containing data objects to be issued to an agent in line 1, the loop in lines 3-7 sifts the database for data objects that satisfy the agent's request to be added to a temporary set, *temp*, initialised in line 2. The data objects in *temp* are then sorted in ascending order of allocation frequency in line 8. Thus, each data object and its allocation frequency are stored in the distributor's database as a key-value pair. The loop in lines 9-3 then appends the agents set with sorted data objects in *temp* in line 11, depending on the quantity of data objects requested. Additionally, the allocation frequency of each data object added to the agent set is incremented by a value of one in line 12.

Transposition Encoding

This study proposes a technique for establishing the originality, hence, ownership of distributed data objects called Transposition Box Encoder (TBE). Specifically, the TBE uses the underlying philosophy of a symmetric key algorithm called transposition cipher to permutate attribute values between tuples, but with a key. The encoding process creates a unique pattern in the dataset that will enable backtracking to the source in case of leakage. The example below illustrates how the TBE works.

Assume a company decides to outsource the records of 10 employees to an agent. One or more columns from the company's database are selected for encoding. For instance, one column whose attribute values are given as $T = \{t_1, t_2, \ldots, t_{10}\}$may be selected. The distributor then chooses a random key, say "xavjtu", for permutation. Note that, longer keys lead to better permutation. The column values are assigned to the characters in the key as shown in Table 2.

Table 2 Unsorted Mappings

Key Chars	a	j	t	u	v	x
Values	t_1	t_2	t_3	t_4	t_5	t_6
	t_7	t_8	t_9	t_{10}		

While maintaining their assigned attribute values, the characters in the keys are sorted alphabetically as shown in Table 3.

Table 3 Sorted mappings

Key Chars	a	j	t	u	v	x
Values	t_2	t_4	t_5	t_6	t_3	t_1
	t_8	t_{10}			t_9	t_7

Finally, a mapping is formed between attribute values in the unsorted Table 2 and sorted Table 3 as shown in Table 4.

Table 4 Transposition Pairs

Original Attributes	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}
Transposed Attributes	t_2	t_4	t_5	t_6	t_3	t_1	t_8	t_{10}	t_9	t_7

The mapping represents a permutation of attribute values that creates unique and hidden pattern in the transposed and outsourced data records known only to the data distributor. If the transposed data records embedded get leaked, tracing the source of the leakage reduces to the task of revealing the hidden patterns in the leaked data records using the appropriate unique keys.

Addition of Fake Objects

The TBE technique is robust enough for leakage detection, but the addition of fake objects where permissible can be used to quickly confirm data leakage source. The fake objects may be created by data owners into a pool fake object. During outsourcing, the list of fake objects associated with datasets assigned to specific agents together with the agents' identities are returned from the pool by a function created by the data owner. Agents typically have a one-to-many relationship with the fake objects. It is important to mention however that, the

addition of fake objects may sometimes impact what agents do; hence their addition may not always be allowed (Papadimitriou and Garcia-Molina, 2011).

2.2.3 Leakage Detection and Guilt Assessment

Leakage Detection

A data leakage suspicion is said to arise when dataset similar to an agent's dataset is detected in an unauthorized location. The suspicious dataset is subjected to leakage detection and guilt assessment tests upon discovery in order to establish its lineage and provenance. During the leakage detection test, records of historical queries on the distributor's database are checked. A comparison is then made between records generated by each query and the suspicious dataset by computing similarity between them. Intuitively, high similarity between the records of query-generated datasets and the records in the suspicious set indicates a high likelihood that data leakage might have indeed occurred. The similarity between dataset allocated to an agent and leaked (suspicious) dataset is computed in terms of a *Detection Rate (DR)* explained below.

Let $T(Ag_x)$ denote dataset allocated to an agent, and $T(S_l)$ denote the suspicious leaked dataset. The similarity between $T(Ag_x)$ and $T(S_l)$ in terms of a Detection Rate denoted by $DR(Ag_x)$, is an intuitive numeric similarity metric that compares the number of data records that are common to the query-generated and the suspicious dataset. Thus, the $DR(Ag_x)$ expressed as a percentage is given by Equation 1.

$$DR(Ag_x) = \frac{\sum_1^{n=|T(S_l)|} {}_{T(Ag_x) \cap T(S_l)}}{\sum_1^{n=|T(S_l)|} {}_{T(S_l)}} x100 \qquad (1)$$

Two records are said to be common if and only if the attributes values of all corresponding fields in the two records are the same. High values of $DR(Ag_x)$ indicate high likelihood that the suspicious dataset might have originated from the data distributor's database.

Guilt Assessment

Having established data leakage, agent guilt assessment is conducted by investigating data lineage and provenance. To establish the data provenance, a composite of three models is utilized namely Probabilistic, Transposition and Fake objects guilt assessments. Each model is explained below.

Probabilistic guilt assessment: This mathematical approach originally propounded by (Papadimitriou and Garcia-Molina, 2009; Papadimitriou and Garcia-Molina, 2011) computes the probability that an agent is guilty of data leakage. Assume a company owns a finite set of $T = \{t_1, t_2, \dots, t_n\}$ records where each $t_i \in T$ is a data record $\forall i = 1, 2, \dots, n$. Let $T(Ag_1) = \{t_1, t_2, t_3, t_5\}$ and $T(Ag_2) = \{t_1, t_4, t_5, t_6\}$ denote the set data records received by agents Ag_1 and Ag_2 respectively, where $[T(Ag_1), T(Ag_2) \subseteq T]$. Also let $T(S_l) = \{t_1, t_2, t_5, t_6\}$ denote a set of suspicious leaked dataset. Let the assumption hold that the data records in the leaked dataset $T(S_l)$ could only have come from Ag_1 or Ag_2. Additionally, assume that an agent's decision to leak a data record is independent of the agent's decision to leak other data records. If the probability that each record in the leaked dataset $t_1 \in T(S_l)$ was obtained by other means is equal to p, it follows that the probability that a record $t_1 \in T(S_l)$ is leaked equal $(1 - p)$. Since the data record t_1 was given to the two agents, the probability that either agent Ag_1 or Ag_2 leaked t_1 is given by

$$P(G_{Ag_1}|t_1) = P(G_{Ag_2}|t_1) = (1 - p)/2$$

where $P(G_{Ag_1}|t_1)$ and $P(G_{Ag_2}|t_1)$ are the respective probabilities that the agents Ag_1 and Ag_2 are guilty of leaking the given the data record t_1. In contrast, the probability that the data record t_2 is leaked is given by

$$P(G_{Ag_1}|t_2) = (1 - p)$$

since only one (agent Ag_1) of the two agents received the said data record. It follows from the ensuing that the probabilities that an agent or the two agents are guilty of leaking the remaining data records are

$$P(G_{Ag_1}|t_5) = P(G_{Ag_2}|t_5) = (1 - p)/2$$

and

$$P(G_{Ag_1}|t_6) = (1 - p)$$

Given the independence assumption made earlier, it can be concluded from the above formulations that, the probability that agent Ag_1 has not leaked any data record is joint probability of the agent not leaking each data record given by Equation 2.

$$P\left(\overline{G_{Ag_1}}\middle|T(S_l)\right) = \left(1 - \frac{1-p}{2}\right) \times \left(1 - (1 - p)\right) \times \left(1 - \frac{1-p}{2}\right) \qquad (2)$$

Consequently, the probability that agent Ag_1 is guilty computes to the relation expressed in Equation 3.

$$P\left(G_{Ag_1}\middle|T(S_l)\right) = 1 - P\left(\overline{G_{Ag_1}}\middle|T(S_l)\right) \quad (3)$$

Similarly, probability that agent Ag_2 has not leaked any data records is given by Equation 4.

$$P\left(\overline{G_{Ag_2}}\middle|T(S_l)\right) = \left(1 - \frac{1-p}{2}\right)\left(1 - \frac{1-p}{2}\right) \quad (4)$$

Hence, the probability that agent Ag_2 is guilty computes to the expression in Equation 4.

$$P\left(G_{Ag_2}\middle|T(S_l)\right) = 1 - P\left(\overline{G_{Ag_2}}\middle|T(S_l)\right) \quad (5)$$

From the formulations above, it can be shown that the probability that a certain agent Ag_x is guilty of leaking a leaked set $T(S_l)$ is given by the general formula expressed in Equation 6.

$$P\left(G_{Ag_x}\middle|T(S_l)\right) = 1 - \prod_{t_i \in T(S_l) \cap T(Ag_x)}\left(1 - \frac{1-p}{V_t}\right) \quad (6)$$

where t_i denotes a record in the leaked dataset, $T(Ag_x)$ is the set of data records allocated to agent Ag_x, V_t is the frequency of the record t_i, and p is the probability that the record t_i is not leaked by any agent.

Transposition Guilt Assessment: Probabilistic guilt assessments give indications of agents that are likely to leak suspicious datasets. Such estimations allow for margin of error, hence may not be sufficient in absolutely identifying culpable agents. In the light of this shortcoming, this study proposes approach, the transposition guilt assessment, for identifying guilty agents with 100% confidence. In the transposition guilt assessment, the keys used to encode datasets during the TBE are applied to decode the leaked dataset. Algorithm 2 illustrates the steps involved in the transposition guilt assessment.

The leaked dataset (*leakSet*) and keys used to encode datasets issued to agents are taken as input.

Algorithm 2: Transposition Guilt Assessment
Require: *leakSet, key*
Ensure: *keyStatus*
1: *recordCount* ← data objects in *leakSet*
2: *DecodedSet* ← Decode(*leakSet, key*)
3: *count* ← *Ø*
4: **for** data object in *DecodedSet* **do**
5: present ← *CompareToDatabase*(data object)
6: **if** present is true **then**
7: *count* ← *count + 1*
8: **end if**
9: **end for**
10: *min* ← *recordCount x Θ* {*Θ* is the threshold value}
11: **if** *count* ≥ *min* **then**
12: *keyStatus* ← *Successful*
13: **else**
14: *keyStatus* ← *Failed*
13: **end if**
14: **return** *keyStatus*

In line 1 of the algorithm, a count of records in the *leakSet* is executed. In line 2, the encoding keys are applied on leaked dataset in order to decode it, and data objects stored in a *DecodedSet* if the leaked dataset is successfully decoded. To ensure that the data objects in the *DecodedSet* match datasets in the distributor's database, the loop in line 3-8 checks for the presence of data objects in the leaked set from the database. If presents, all matching objects are counted and compared with the minimum threshold value, *min*, set in line 9. Lines 10-14 check if the matching count is at least up to the threshold value. If the count passes the threshold, the key is declared *Successful* else it is declared *Failed*.

It is worthy of note that the choice of the min threshold value is dependent on data type involved. Threshold values for very sensitive datasets are set low since these types of datasets are very unique and nearly impossible to be any source other that the distributor's database. For most purposes, the threshold value is set to 0.4 for sensitive datasets and 0.7 for other data types.

Fake Objects Guilt Assessment: The presence of one or more fake objects can also be used in identifying leaked data source. When leakage occurs, the suspicious leaked dataset is collected and fake record test performed. The test returns the detected fake object IDs and the ID of the agent who received the fake objects. Algorithm 3 details the fakes objects detection test procedure.

Algorithm 3: Fake Object Detection
Require: *leakSet*
Ensure: *DetectedList*
1: **for** record in leakSet **do**
2: **if** recordID is a FakeID **then**
3: DetectedList ← recordID
4: **end if**
3: **end for**
4: **return** DetectedList

The algorithm takes the leaked dataset as its input and returns the list of fake records found in the leaked set. The loop in lines 1-4 compare the IDs of data objects in the leaked data set to fake objects contained in the distributor's database. If the IDs match, the object's ID is added to a list of detected fake objects (*DetectedList*). For the guilt assessment, the *DetectedList* is compared with the fake object dataset issued to agents.

3 Results and Discussion

3.1 Experimental Setting

To validate the proposed leakage detection approach, the hypothetical data allocation scenario described Section 1 is considered. More specifically, the data leakage detection problem is tackled using the three (3) systematic steps listed below:

(i) Allocation of datasets to agents;
(ii) Tests run on the suspicious datasets to prove data ownership by a distributor and to detect the occurrence of data leakage; and
(iii) Establishment of provenance following the leakage detection above.

3.1.1 Allocation of Datasets to Agents

The three agents were allocated dataset from 300 records of the AER credit card using the data allocation Algorithm 1. Ag_1 received $T(Ag_1) =$ 200 records; agent Ag_2, $T(Ag_2) = 150$ records; and agent Ag_3, $T(Ag_3)$ records. Having allocated datasets, the following overlapping or identical records between datasets received by Ag_1, Ag_2 and Ag_3 were identified:

$$\text{(i)} \quad n\big(T(Ag_1) \cap T(Ag_2)\big) = 50$$
$$\text{(ii)} \quad n\big(T(Ag_1) \cap T(Ag_3)\big) = 150$$
$$\text{(iii)} \quad n\big(T(Ag_2) \cap T(Ag_3)\big) = 100$$

where $n\big(T(Ag_1) \cap T(Ag_2)\big)$, $n\big(T(Ag_1) \cap T(Ag_3)\big)$ and $n\big(T(Ag_2) \cap T(Ag_3)\big)$ are respectively the number of overlapping records between Ag_1 and Ag_2; $Ag_1$and Ag_3; as well as $Ag_2$and Ag_3.

For the purposes of the experiment conducted in this study, 3 and 2 fake objects were respectively added to the data records received by Ag_1 and Ag_2. Thus, in total Ag_1, Ag_2 and Ag_3 received 203, 152 and 250 data records respectively. Fig. 3 exemplifies the data allocation process to an agent.

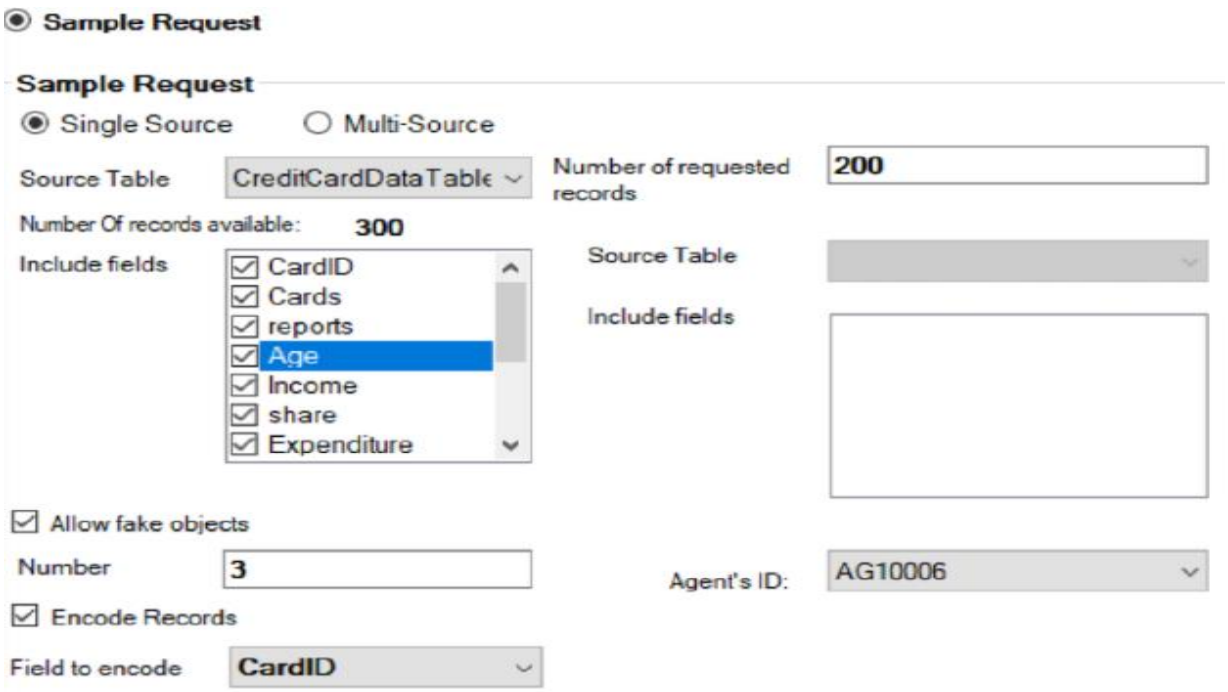

Fig. 3 Agent Data Allocation Process

3.1.2 Leakage Detection Test

It is assumed that the dataset allocated an agent Ag_1were leaked. Thus, the leaked dataset denoted by $T(S_l)$, is $T(S_l) \equiv T(Ag_1)$ = 203 which suggests that all the dataset allocated to Ag1 is the same as the leaked dataset. Given that

$$\sum_{1}^{n=|T(S_l)|} T(Ag_1) \cap T(S_l) = 200$$

In accordance with Equation 1, the detection rate $DR(Ag_1)$ becomes

$$DR(Ag_1) = \frac{200}{203} x100 = 98.5\%$$

Similarly,

$$\sum_{1}^{n=|T(S_l)|} T(Ag_2) \cap T(S_l) = 50$$

Hence, the $DR(Ag_2)$ is calculated as

$$DR(Ag_2) = \frac{50}{203} x100 = 24.6\%$$

Additionally,

$$\sum_{1}^{n=|T(S_l)|} T(Ag_3) \cap T(S_l) = 150$$

Hence, the $DR(Ag_3)$ is calculated as

$$DR(Ag_3) = \frac{150}{203} x100 = 73.8\%$$

Fig. 4 shows the results of test on the credit card-containing database used in the experiments where QU1033, QU1034 and QU1035 denote the queries that produced data for the agents Ag_1, Ag_2 and Ag_3 respectively.

Observe that the detection rate for Ag_1 whose allocated dataset is the same as the suspicious dataset is 98.5%. This falls short of the expected 100% detection rate. This discrepancy is attributable to the 3 fake objects added to Ag_1's dataset. The detection rate for Ag_1, Ag_2 and Ag_3 without considering the addition of fake objects respectively are:

Query ID	Objects Detected (%)
QU1018	0.0
QU1019	0.0
QU1020	0.0
QU1021	0.0
QU1022	0.0
QU1023	0.0
QU1024	0.0
QU1025	0.0
QU1026	0.0
QU1027	0.0
QU1028	0.0
QU1029	0.0
QU1030	0.0
QU1031	0.0
QU1032	98.5
QU1034	24.6
QU1035	73.9

Fig. 4 Results on Detection Test

$$DR(Ag_1) = \frac{200}{200} x100 = 100\%$$

$$DR(Ag_2) = \frac{50}{200} x100 = 25\%$$

$$DR(Ag_3) = \frac{150}{200} x100 = 75\%$$

The discrepancy between the detection rate with fake objects and detection rate without fake object is known as detection error, ε. Where the necessity of adding fake objects arises, it should be done so as to minimize ε as much as possible. ε with respect to an agent Ag_x is given by Equation 7.

$$\varepsilon(Ag_x) = |DR(Ag_x) - DR(Ag_x)'| \quad (7)$$

where $DR(Ag_x)'$ is ε of an agent Ag_x with no fake objects added. $\varepsilon(Ag_1)$, $\varepsilon(Ag_2)$ and $\varepsilon(Ag_3)$ for agents Ag_1, Ag_2 and Ag_3 are respectively 1.5%, 0.04% and 1.1%.

3.2 Agent Guilt Assessment

An agent guilt assessment is a post-detection forensics conducted to trace the agent responsible for the leak after data leakage has been confirmed.

3.2.1 Probabilistic Guilt Assessment

The probabilistic guilt assessment is based on Equation 6 and conducted using datasets allocated to Ag_1, Ag_2 and Ag_3. Fig. 5 shows the results obtained from experiments for each agent.

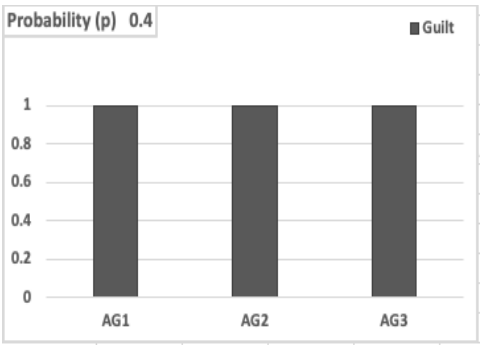

Fig. 5 Assessment of 200 Leaked Objects

Note that a probability value of 0.4 is set as minimum threshold value of culpability. The probability values of 1.0, 0.99999, 0.99999 obtained represent estimated likelihood of guilt levels of agents Ag_1, Ag_2 and Ag_3 respectively. However, with these similar probability values for the three agents, it makes it nearly impossible to identify the culpable agent using this approach.

The observation above is because, if the number of data objects common to dataset allocated to an agent and the suspicious dataset exceeds a certain threshold, the outcome of Equation 6 is almost always guaranteed to be approximately equal to 1. The product term in Equation 6 which represents the joint probability that an agent is not guilty of leaking the data objects approaches zero (0) as the number of leaked data objects increase.

It needs to pointed out that, with smaller sizes of leaked data objects, the probabilistic guilt assessment technique is able to clearly identify without much room for doubt, the culpability of an agent. The evidence of this claim is illustrated in Fig. 6 which is an experiment conducted by randomly sampling 10 data objects ear-marked as leaked dataset.

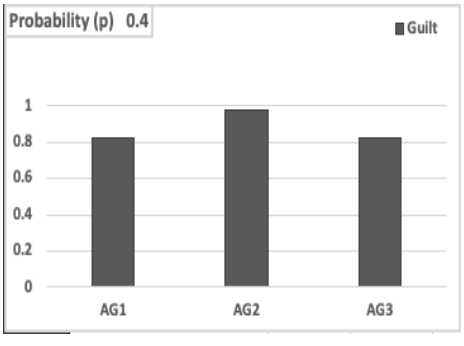

Fig. 6 Assessment of 10 Leaked Objects

With a minimum probability threshold value set at 0.4, the three agents Ag_1, Ag_2 and Ag_3 exhibit high probability values of 0.82, 0.98 and 0.82 respectively. With these values, it is easy to pin-point agent Ag_2 as the culpable agent since the estimated likelihood of leaking the data objects for this agent is the highest.

3.2.2 Fake Object Guilt Assessment

Since fake objects are uniquely allocated to agents, hence the presence of a particular fake object puts the owner of that object in the guilt segment. Fig. 7 exemplifies the results of fake object guilt assessment conducted using the AER Credit Card Data allocated to the three agents earlier.

Run Test

Object ID	Agent ID	Status
UT02001	AG1	Detected!!!
UT02002	AG1	Detected!!!
UT02003	AG1	Detected!!!

Fig. 7 Fake Object Guilt Assessment

It can be seen that fake objects detected belong to the agent with id AG1 designated agent Ag_1 earlier. This points to an indisputable culpability of the agent Ag_1 for the leak which is consistent with the expected outcome.

Note that even though fake object guilt assessment may lead to conclusive outcomes, sometimes conditions accompanying agents' requests may make it impossible to add fake objects to allocated datasets. The proposed transposition key guilt assessment can be employed whether or not there exists a fake object pool within the allocated datasets to conclusively identify a guilty agent.

3.2.3 Transposition Guilt Assessment

As discussed earlier, the key used to encode datasets allocated to each agent during the transposition encoding phase is unique, and facilitates the identification of guilty agent(s) with 100% confidence. In the experiment, datasets issued to agents were encoded by the transposition encoder Algorithm 2 each with a unique key. The results obtained from the transposition key guilt assessment based on the leakage scenario mentioned is shown in Fig. 8.

The key with the id Boatewxfy was the only key successful in decoding the data. This suggests that the owner of the key is the agent responsible for the leakage. From the system, the key can only be linked to the agent with id AG1 who is agent Ag_1 in our scenario. This approach corroborates the outcome from the experiment conducted using the fake object guilt assessment approach.

Run	
Key ID	**Outcome**
Afridjcec	Failed
Agbeolkbv	Failed
Agbeyvevy	Failed
Akakfqcqt	Failed
Akakrdalc	Failed
Aswefvdsm	Failed
Boatardrh	Failed
Boatewxfy	Successful
Dolewedss	Failed
Felisfxjv	Failed
Gakukihol	Failed

Fig. 8 Transposition Key Test

The main strength of the proposed transposition guilt assessment lies in the fact that, besides being able to identify culpable agents with 100% confidence, the outcomes of its experiments are agnostic of the size leaked data objects involved.

4 Conclusions and Recommendations

This study proposes a robust approach for data leakage and guilt assessment problem. In particular, a multi-faceted data allocation strategy is developed which includes a novel data transformation technique known as transposition encoding that facilitates leakage detection and guilt assessment. The proposed transposition encoding technique addresses the shortcomings of the traditional watermarking approaches which include perturbation of data attribute before allocation. The proposed transposition guilt assessment technique is unique in that it can identify with 100% confidence, a culpable agent responsible for dataset leakage. For future works, it is recommended that the transposition encoder algorithm be extended to encode correlated attributes in a relational database such as name and email, age and date of birth, etc.

References

Agrawal, R. and Kiernan, J. (2002) "Watermarking relational databases", *In Proceedings of the 28th International Conference on Very Large Data Bases*, *VLDB '02, VLDB Endowment*, Hong Kong, China pp. 155–66.

Al Solami, E., Kamran, M., Saeed Alkatheiri, M., Rafiq, F. and Alghamdi, A. S. (2020), "Fingerprinting of Relational Databases for Stopping the Data Theft". *Electronics*, Vol. 9, No. 7, pp.1093.

Buneman, P., Khanna, S. and Wang-Chiew, T. (2001), "Why and where: A characterization of data provenance", *in J. Van den Bussche and V. Vianu, eds, Database Theory - ICDT 2001*, Springer, Berlin, Heidelberg, pp. 316–330.

Chanvarasuth, P. (2008), "The impact of business process outsourcing on firm performance", *in 'Fifth International Conference on Information Technology: New Generations*', Las Vegas, NV, pp. 698–703.

Cheney, J., Chiticariu, L. and Tan, W.-C. (2009), "Provenance in databases: Why, how, and where", *Found. Trends databases*, Hanover, MA, USA Vol. 1, No. 4, pp. 379–474.

Davis, R. I. and Burns, A. (2007), "Robust priority assignment for fixed priority real-time systems", *28th IEEE International Real-Time Systems Symposium (RTSS 2007)*, Tucson, AZ, pp. 3–14.

Espino-Rodríguez, T.F. and Ramírez-Fierro, J.C. (2017), "Factors determining hotel activity outsourcing. An approach based on competitive advantage", *International Journal of Contemporary Hospitality Management*, Vol. 29 No. 8, pp. 2006-2026

Greenberg, P., Greenberg, R. and Antonucci, Y. (2008), "The role of trust in the governance of business process outsourcing relationships: A transaction cost economics approach" *Business Process Management Journal*, Vol. 14. pp. 593-608

Guo, F., Wang, J., Zhang, Z., Ye, X. and Li, D. (2006), "An improved algorithm to watermark numeric relational data", *Proceedings of the 6th International Conference on Information Security Applications*, WISA'05, Springer-Verlag, Berlin, Heidelberg, pp. 138–149.

Kamran, M. Z. and Farooq, M. (2018), "A comprehensive survey of watermarking relational databases research", *https://deep-ai.org/publication/a-comprehensive-survey-of-watermarking-relational-databases-research*. Accessed October 12, 2019

Katcher, D. I., Arakawa, H. and Strosnider, J. K. (1993), "Engineering and analysis of fixed priority schedulers", *IEEE Transactions on Software Engineering Journal* Vol 19, No. 9, pp. 920-934.

Khan, N., Yaqoob, I., Hashem, I., Inayat, Z., Kamaleldin, W., Alam, M., Shiraz, M., and Gani, Abdullah (2014), "Big Data: Survey, Technologies, Opportunities, and Challenges", *https://www.readcube.com/articles/10.1155/2014/712826*. Accessed October 12, 2019

Koti, B.R., Kumar, G.R. and Srinivas, Y. (2017), "A comprehensive study and comparison of various methods on data leakages", *International Journal of Advanced Research in Computer Science*, Vol. 8, No. 7, pp. 627-631.

Kowalczyk, M. and Buxmann, P. (2014), "Big Data and Information Processing in Organizational Decision Processes: A Multiple Case Study", *Business & Information Systems Engineering*, pp. 267–278.

Mazumdar, S., Seybold, D. and Kritikos, K. (2019), "A survey on data storage and placement methodologies for Cloud-Big Data ecosystem", *Journal of Big Data*, Vol 6, No. 15, pp. 1-37.

Papadimitriou, P. and Garcia-Molina, H. (2009), A model for data leakage detection", *2009 IEEE 25th International Conference on Data Engineering*, Shanghai, China, pp. 1307–1310.

Papadimitriou, P. and Garcia-Molina, H. (2011), "Data leakage detection", *IEEE Transaction on Knowledge and Data Engineering*, USA, Vol. 23, No. 1, pp. 51–63.

Radu Sion, M. A. (2004), "Attacking digital water-marks", *Proceedings of SPIE - The International Society for Optical Engineering*, San Jose, California, Vol. 5306, pp. 848-858.

Ram, J., Zhang, C. and Koronios, A. (2016), "The Implications of Big Data Analytics on Business Intelligence: A Qualitative Study in China", *Procedia Computer Science*, pp. 221-226. 10.1016/j.procs.2016.05.152.

Schulze, H. (2018), "Insider threat 2018 report", *https://www.africacybersecurityconference.com /document/Cyber%20Security-insider-threat-report.pdf*. Accessed December, 2018.

Shabtai, A., Elovici, Y. and Rokach, L. (2012), "A survey of data leakage detection and prevention solutions", *Springer Briefs in Computer Science*, Springer Publishing Company, Incorporated, Springer, Boston, MA., pp. 17-37.

Shehab, M., Bertino, E. and Ghafoor, A. (2008), "Watermarking relational databases using optimization-based techniques", *IEEE Transactions on Knowledge and Data Engineering Journal*, Vol. 20, No. 1, pp. 116–129.

Sion, R., Atallah, M. and Prabhakar, S. (2003), "Rights protection for relational data", Proceed*ings of the 2003 ACM SIGMOD International Conference on Management of Data*, ACM, New York, NY, USA, pp. 98–109.

Sion, R., Atallah, M. and Prabhakar, S. (2004), "Rights protection for relational data", *IEEE Transaction on Knowledge and Data Engineering Journal* Vol. 16, No. 2, pp. 1509–1525.

Zhang, Z.-H., Jin, X.-M., Wang, J.-M. and Li, D. Y. (2004), "Watermarking relational database using image", *Proceedings of 2004 International Conference on Machine Learning and Cybernetics*, (IEEE Cat. No.04EX826), Shanghai, China, Vol. 3, pp. 1739–1744.

Authors

H. Abdel-Fatao is a Lecturer in the Computer Science and Engineering Department at the University of Mines and Technology, Tarkwa, Ghana. He holds a PhD in Computer and Information Sciences from the University of South Australia, Adelaide. His research interests include GPS trajectory data mining, artificial intelligence, robotics and embedded computing, software and web development, bio-informatics.

V. M. Nofong is a Lecturer at the University of Mines and Technology, Tarkwa, Ghana. He received a Ph.D. degree in Computer and Information Science in 2016 at University of South Australia, Adelaide and a B.Sc. degree in 2010 from the University of Mines and Technology, Tarkwa, Ghana. His current research interests include data mining, pattern mining, classification and trend prediction.

L. K. Agbeyeye has his bachelor's degree in Computer Science and Engineering from the University of Mines and Technology, Tarkwa, 2018. His is currently pursuing his Masters degree in Computer Science and Software Engineering at the Schaffhausen Institute of Technology. His research interests lie in software quality and machine learning.

M. Y. Umaru is an Assistant Lecturer in the Computer Science and Engineering Department at the University of Mines and Technology, UMaT, Tarkwa. He is currently pursuing PhD in Computer Science and Engineering at UMaT, Tarkwa. His research interest is in design, implementation, and analysis of resource restricted embedded real-time systems. His current is looking at the implementation of Real-Time monitoring system on hardware platforms.

A. K. Alese is currently a Professor with the Computer Science Department of the Federal University of Technology Akure, Ondo State, Nigeria. He holds a Ph.D. degree in Computer Science from The Federal University of Technology Akure, Ondo State, Nigeria. His areas of research include, Computer and Network Security, Quantum Computing and Digital Signal Processing.