1-1-2022

Secure Storage Model for Digital Forensic Readiness

Avinash Singh
University of Pretoria

Richard Adeyemi Ikuesan
Zayed University

Hein Venter
University of Pretoria

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.Doi Number

Secure Storage Model for Digital Forensic Readiness

Avinash Singh[1], Richard Adeyemi Ikuesan[2], and Hein Venter[1]

[1] University of Pretoria, Pretoria, South Africa

[2] College of Technological Innovation, Zayed University, Abu Dhabi, UAE.

Corresponding author: Avinash Singh (e-mail: asingh@cs.up.ac.za).

ABSTRACT Securing digital evidence is a key factor that contributes to evidence admissibility during digital forensic investigations, particularly in establishing the chain of custody of digital evidence. However, not enough is done to ensure that the environment and access to the evidence are secure. Attackers can go to extreme lengths to cover up their tracks, which is a serious concern to digital forensics – particularly digital forensic readiness. If an attacker gains access to the location where evidence is stored, they could easily alter the evidence (if not remove it altogether). Even though integrity checks can be performed to ensure that the evidence is sound, the collected evidence may contain sensitive information that an attacker can easily use for other forms of attack. To this end, this paper proposes a model for securely storing digital evidence captured pre- and post-incident to achieve reactive forensics. Various components were considered, such as integrity checks, environment sandboxing, strong encryption, two-factor authentication, as well as unique random file naming. A proof-of-concept tool was developed to realize this model and to prove its validity. A series of tests were conducted to check for system security, performance, and requirements validation, Overall, the results obtained showed that, with minimal effort, securing forensic artefacts is a relatively inexpensive and reliable feat. This paper aims to standardize evidence storage, practice high security standards, as well as remove the need to create new systems that achieve the same purpose.

INDEX TERMS Digital Forensic Readiness; Secure Storage; Integrity Verification; Encryption, Digital Forensic Soundness

I. INTRODUCTION

The upsurge in cyber-attacks and data exploitation has made the need for digital investigations paramount [1]–[3]. Standardization and adherence to best practices have become essential to ensure the least amount of human error causing inadmissible evidence [4], [5]. Forensic artefacts are very important when it comes to investigation and litigation, as they provide the details of an incident [6], [7]. When forensic artefacts are presented in a court of law, they are subject to scrutiny and require verification and cross-examination [8], [9]. Digital evidence needs to preserve the CIA triad [10], namely confidentiality, integrity, and availability. Confidentiality of digital evidence must be ensured because the evidence may contain sensitive information such as credit card information or other personal identifiers [10]. To protect the evidence, strict access control needs to be maintained, and/or an encryption scheme has to be used to ensure that only an investigator or authorized parties have access to the digital evidence [11]. Ensuring the integrity of the digital evidence is one of the most important processes of any digital investigation, as an investigator needs to prove that the evidence was not fabricated or tampered with in any way. To achieve this, a forensic copy of the original evidence, as well as the software logs and the chain of custody, is kept [12]. The process followed by the investigator to acquire the evidence also needs to be documented. The forensic hash of the evidence needs to be calculated at different times – during the time of collection and storage – to ensure that the original evidence was not changed, and the process followed by the investigator was sound and did not modify the evidence in any way. Therefore, a secure storage model is needed to improve the investigation process and safeguard any sensitive information collected. The same problem affects digital forensic readiness systems, whether large or small organizations or even individual people. These systems collect evidence proactively, therefore, evidence preservation and storage processes are vital to ensure that evidence is valid and authentic.

The next section provides some background on digital forensic readiness and encryption. Thereafter, the proposed process model is explained, detailing each of the processes involved, followed by the proof-of-concept prototype tool that was developed. Next, the tool was evaluated in terms of usefulness and performance, before the paper is concluded.

II. BACKGROUND

Digital forensic readiness (DFR) as defined by Tan [13] is the ability of an organization to maximize its evidence collection mechanisms whilst aiming to reduce the costs involved in collection [13]. Therefore, to achieve DFR, potential digital evidence collection needs to take place before an incident can occur. DFR is a proactive approach to digital forensics that is more robust and cost-effective in the long term. To implement DFR in any organization, its business operations need to be well defined and understood, as they may differ from organization to organization. The ISO/IEC 27043 international standard [14] defines a more robust guideline about the traditional digital investigation processes as well as high-level readiness processes. This encompasses five processes, namely readiness, initialization, acquisition, investigative, and concurrent processes [14]. It also ties in with the investigation lifecycle as shown in Figure 1, which consists of planning, acquisition, preservation, analysis, reporting, dissemination, and chain of custody. Most research focuses on the acquisition and analysis of evidence; however, little is done about the preservation of evidence and its integrity. No comprehensive models or guidelines are defined for evidence integrity preservation, specifically in digital forensic readiness. Although ISO/IEC 27037 [15] contains a clause on evidence preservation that outlines general guidelines on the physical storage and preservation of evidence, it is not sufficient for comprehensive evidence integrity preservation in terms of storage security.

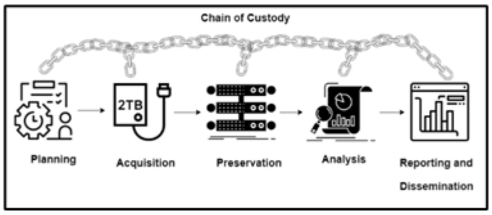


Figure 1. Investigation Lifecycle

Most existing research focuses on using encryption to secure data that is being stored [16]–[19]. Due to the nature of the data stored, most common encryption focuses on symmetric encryption which means that there is a single encryption key that also serves as the decryption key [11], [20]. AES (Advanced Encryption Standard) is the dominant (standard) encryption scheme used by cloud and enterprise platforms because of its speed and performance [11], [20], [21]. However, if the decryption key is not stored securely, it leaves the encrypted data still vulnerable to be stolen or misused. At the time of writing this paper, no model or framework provides the best practice on how to securely store data and ensure its integrity in a digital forensic environment. In digital forensics, it is essential that the integrity of the data remains intact to make it reputable and admissible in a court of law [22], [23]. In DFR, potential digital evidence (PDE) is defined as information or data stored or transmitted in binary form, which has not yet been determined to be relevant to the investigation (through the process of examination and analysis) [14]. Only after the PDE has been positively identified as evidence, it is accepted as digital evidence. To mimic a more real-world application of PDE with DFR in mind all references to PDE are made to simplify the explanations. PDE can be seen as small artefacts (not large disk dumps) that may hold important or sensitive data. Since PDE could be used to incriminate an individual, it needs to maintain its integrity and authenticity to be admissible. Therefore, some processes must be in place to ensure the correct steps and processes are followed, to ensure the integrity of the PDE. To date, no processes or models exist to address the integrity constraint, especially with DFR. There are also no tools that focus solely on the storage of PDE. On the contrary, these tools focus on the extraction and collection of PDE, and it is up to the investigator or organization to safeguard and preserve the PDE according to their policies [24].

Anti-forensics is cumbersome in cloud environments, and attackers are always trying to cover up their footprints [25]. They usually move laterally in a network to find vulnerabilities and exploit them by removing any traces of the attack from the logs and computers. Therefore, having a secure environment is important, and sensitive information should be secured and encrypted. While several cloud service providers do provide encryption, it often comes at a huge cost or additional overhead and attackers can easily bypass the service providers' countermeasures by targeting less secure Virtual Machines (VMs) [25]. While several studies have explored readiness in the cloud [15], [26]–[28], the more fundamental problem is providing secure storage for the PDE artifacts that are collected. To that end, this research developed a secure storage system to store digitally forensic ready PDE artifacts in a forensically sound manner. The next section explains the developed process model, namely Secure Readiness Storage (SecureRS), in accordance with security standards and the digital forensic investigation lifecycle.

III. SecureRS PROCESS MODEL

To address the lack of an automated mechanism for preserving evidence and maintaining integrity, a model was developed targeting the various security and forensic aspects during the investigation lifecycle. This model is an improvement of the authors' previous work [29]. The

SecureRS model ties in with some of the readiness processes addressed in ISO/IEC 27043 [14]. For instance, the planning process group of ISO/IEC 27043 [14] involves the "Planning pre-incident collection, storage, and handling of data representing potential digital evidence" [14]. It discusses the criteria of collection and storage, but nothing is provided on how storage and evidence preservation should be carried out. Therefore, the proposed model consists of four high-level processes, namely data ingestion, forensic soundness assurance, PDE storage, and forensic soundness verification (see Figure 2).

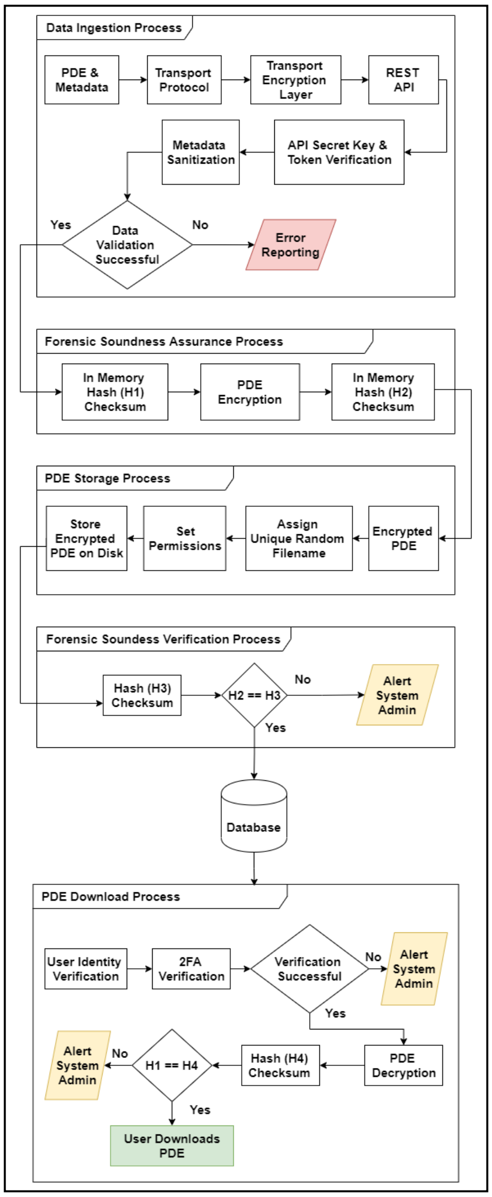


Figure 2. SecureRS Process Model

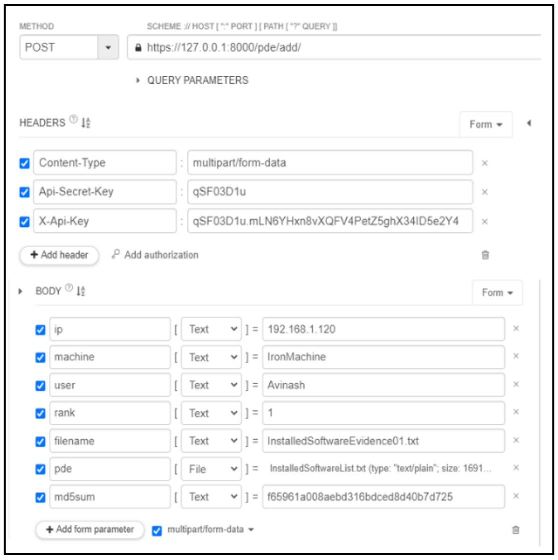
The data ingestion process acts as a mechanism for data to be fed into the secure system. It ensures a controlled environment because it is common practice to make use of a Web API. The SecureRS model makes use of a Representational State Transfer (REST) API to allow a multitude of data ingestion formats as well as a consistent endpoint with a lower bandwidth than other API types. When questioning the integrity of a storage engine or system, it is important to understand what processes the data undergoes. To ensure that the system or a user does not modify any information, integrity checksums are calculated before and after to ensure nothing was changed. Since the collection of evidence does not fall within the scope of the current research, it is assumed that data collected and sent to the model is forensically sound already. The PDE storage process is built for security, based on the CIA triad. During the final stage – the verification process – the integrity before and after storage is checked to ensure nothing has been modified, thus ensuring forensic soundness. The four processes followed in the model are discussed next in more detail. The four phases are outlined, and how each aids the security as well as forensic aspects that a piece of evidence needs to possess for the evidence to be admissible in a court of law.

A. DATA INGESTION PROCESS

The data ingestion process comprises seven steps as seen in Figure 2. This process starts with Potential Digital Evidence (PDE) (i.e., small artefacts or pieces of data that may have forensic value) and the PDE's metadata, which helps the system identify the origin and purpose of the PDE. The collection of PDE is not considered part of the scope of this research, as it is a vast research area on its own and only the storage processes are explored in this research. PDE metadata that is needed includes the user or origin, IP address, computer name, rank, file name, and hash checksum. This information is necessary, particularly in an organization, to know where the PDE originates from and to manage the data.

The next stage involves using a transport protocol so that the data from the origin can be received by the system. Different protocols are available; however, the most used and common transport protocol, which is the foundation of the internet, is TCP/IP. Using the TCP/IP method for data transport makes it reliable for data ingestions. The transport method also needs to be secured to prevent eavesdroppers or man-in-the-middle attacks. The transport encryption layer that is chosen involves making use of the secure socket layer (SSL). This SSL layer, coupled with TCP/IP together with HTTP and its application layer protocol, provides HTTPS. An HTTPS connection provides a secure means of communication that is encrypted between two parties, namely the client and server. Using TLS (Transport Layer Security)/SSL is an industry best practice and standard as a move towards a more secure internet. If data is intercepted (by a man-in-the-middle attack, for example), it will be potentially unusable to an attacker as all data would be encrypted [30]. In the SecureRS model, it was decided to make use of a web REST API for data transfer and logic processing. Web APIs are portable and the most modular method of easily ingesting data, requiring minimal effort to set up. To make the ingestion process faster and more standardized, a known standardized API endpoint (similar to a URL path) is exposed on the webserver for data to be ingested. Furthermore, allowing only the HTTP POST method ensures that data remains secure and encrypted in the body of the request. This method also allows large files to be sent – as opposed to the HTTP GET method. The data transferred in the POST method prevents the webserver from logging the request information as what can be seen from GET requests in server logs. Such webserver logging could prove harmful as the GET request parameters would be passed through the URL, which may contain sensitive information. For simplicity, the data format that is supported for ingestion is form data, as this allows both textual information and file upload.

To ensure that only authorized parties can push data to the storage engine, an API key and prefix key are generated through a system admin account for each device/user within the organization. The API key is hashed before it is stored, therefore, the key is only displayed and available at the time it is created. The API prefix furthermore serves as a unique identifier. The API token is used in conjunction with the prefix key to add another layer of security. The prefix key and API key are then verified and, once successful, the metadata is sanitized. This sanitization removes any malicious data, SQL, or JS injections from cross-site scripting (XSS) (from the metadata only). The PDE itself is treated as a read-only file and not displayed in the system, thereby removing the need to perform any sanitization, and so ensuring the integrity of the PDE. The metadata sanitization is performed to ensure that no exploits and vulnerabilities are exposed by the system itself and to conform to best web security practices. The metadata collected about the PDE is shown in Figure 3, which is kept separate from the PDE. After data has been sanitized to ensure system security, the next process is data validation.



```
POST  https://127.0.0.1:8000/pde/add/

HEADERS
Content-Type : multipart/form-data
Api-Secret-Key : qSF03D1u
X-Api-Key : qSF03D1u.mLN6YHxn8vXQFV4PetZ5ghX34ID5e2Y4

BODY
ip        [ Text ] = 192.168.1.120
machine   [ Text ] = IronMachine
user      [ Text ] = Avinash
rank      [ Text ] = 1
filename  [ Text ] = InstalledSoftwareEvidence01.txt
pde       [ File ] = InstalledSoftwareList.txt (type: "text/plain"; size: 1691...
md5sum    [ Text ] = f65961a008aebd316bdced8d40b7d725
multipart/form-data
```

Figure 3. **SecureRS http request example**

The validation process ensures that the data expected is the data received and that the data is parsed with the correct data structure and format. Once the data has been successfully validated, it gets sent on to the next phase for forensic assurance.

B. FORENSIC SOUNDNESS ASSURANCE PROCESS

For digital evidence to be forensically sound and to be held admissible in a court of law, its collection, storing, and analysis must be documented in a legally acceptable manner [16], [17], [31]. Therefore, assurance is needed to prove that the evidence has not been corrupted or destroyed during the investigation process, whether by accident or intentionally. This process furthermore generates the relevant information (such as hash checksums) to prove the forensic soundness of the collected data once the data has been received. Since this system is simply a storage engine, it is assumed that the data that was collected before the system ingestion was forensically sound. However, since nothing has been written to disk or the database as yet, this process is done in-memory. This is to ensure the data was not modified while being written to the disk, whether by another process or due to human error.

The ability to ensure forensic soundness is achieved by taking an in-memory hash (H1) of the PDE using an MD5 hash algorithm as an integrity measure. It is then compared to the metadata md5sum field received from the HTTP POST request to ensure the data sent by the origin is indeed what is received by the API. This serves a dual purpose, namely, to ensure data was not lost or intercepted along the way, and to maintain the integrity of the PDE. Although MD5 is typically seen as an insecure hashing algorithm, it is very suitable as an integrity measure due to its efficiency in computing a hash as opposed to securing hashing algorithms. The next step is to secure the PDE by performing symmetric key encryption. This is to ensure that data stored on the disk is not subject to being read by another system or person, as a PDE file could contain sensitive information. After PDE encryption, another hash (H2) is generated of the encrypted PDE. This new hash is used as input to the process of forensic soundness verification. The next process involves the storage of the encrypted PDE to disk.

C. PDE STORAGE PROCESS

The storage process starts by taking the encrypted PDE from memory and generating a unique filename of 60 alpha-numeric characters to ensure that the system is immune to URL manipulation. This unique filename prevents a PDE from being easily identified by a system admin since there would just be random encrypted files. The PDE is stored to disk with read-only permissions on the file system, such that no process or human error can accidentally change it, thereby violating the PDE's integrity. After setting the permissions, the file is now safely stored in the secure storage within the protected directory in the virtual environment, ready for verification and integrity confirmation. Changing the extrinsic metadata of the PDE (such as the file name or permissions) does not change the data of the PDE itself, hence the hash and integrity remain intact [18], [32]. Details of the forensic verification and assurance process are presented next.

D. FORENSIC SOUNDNESS VERIFICATION PROCESS

This process involves ensuring that the integrity of the evidence is indeed intact and unaltered, thus adhering to standard forensic practice. To ensure that the integrity of the stored PDE remains intact, a hash (H3) is computed on the stored and encrypted PDE. This adds a verification layer which ensures that the forensic soundness of the PDE is maintained from the point of encryption to the storage of the PDE.

To ensure forensic soundness, the in-memory hash of the encrypted PDE (H2) is then compared to (H3). If H2 and H3 are the same, no deliberate or accidental manipulation of the PDE occurred, and it is verified as forensically sound. When the verification was successful, the entry is inserted into the database for reference. This entry contains the metadata of the PDE itself, such as the location of the stored PDE, and not the actual PDE itself. Storing a reference to a file location in a database – as opposed to storing the entire file – conforms to best practices, due to the inefficiency of storing binary data in a relational database [19]. This also makes it difficult for an attacker as it expands the attack vector by abstracting the PDE itself from the metadata. For example, if an attacker gets unauthorized access to the database, the only information that can be extracted is the metadata which on its own does not give enough information for malicious intent. To further protect the entry in the database, the original hash and PDE location are encrypted by the system, adding a layer of security, and thereby making it impossible for an attacker or system admin to relate PDE to its metadata outside the system. If the hashes in the verification process are not identical, it can be assumed that external modification could have occurred or that something unconventional originated, such as electricity spikes or bad disk sectors, thus resulting in invalidating the forensic soundness. Such an instance would be rare and uncontrolled, and a system admin would be notified to manually investigate what the cause could have been. This investigation is a manual process, as the violation would have occurred under unknown circumstances and was not part of the scope of this research.

E. PDE DOWNLOAD PROCESS

The downloading of PDE is also an important aspect of the system to ensure that only authorized parties are allowed to download the PDE. To protect the PDE, the system first verifies the session of the logged-in user and then prompts the user for the 2FA pin. Once the pin and the session have been successfully validated, then only does the system decrypt the PDE. Thereafter, another hash (H4) is generated and then compared to the original hash (H1) to ensure that nothing has happened to the PDE during storage as well as to verify the integrity of the forensic copy that will be downloaded by the investigator. In the event the hashes do not match, the system will alert the administrator to manually investigate the issue. Therefore, the downloaded PDE that the investigator will receive is safe and its integrity is maintained from ingestion into the system to download, thereby minimizing any human error that can occur as well as serving as a secure backup to PDE. The hash is also given to the investigator if further corroboration or verification is needed. The next section discusses the reference implementation of the proposed SecureRS model.

IV. SecureRS TOOL

To show that the proposed SecureRS process model (see Figure 2) would work and is valid, a proof-of-concept tool was created using agile software development methodology [33]. The requirement specification and usability are the core functions for any software following agile principles. To ensure that the proposed SecureRS proof-of-concept tool adheres to standards and good software practice and principles, the tool was tested using the testing processes of the Computer Forensics Tool Testing (CFTT) program [34] of the National Institute of Standards and Technology (NIST) [35].

A. SecureRS SYSTEM REQUIREMENTS SPECIFICATION

The system requirements are divided into two categories, namely secure storage core requirements (SS-CR) (see Table 1) and secure storage optional requirements (SS-OR) (see Table 2). For example, in Table 1 the label column provides a reference number which will be used in Section C. The description, on the other hand, provides the requirements for the tool, for example, SS-CR-01 says that the tool shall ingest data from an API endpoint, which specifies the functionality of the tool.

Table 1. Secure Storage Core Requirements (SS-CR)

Label	**Description**
SS-CR-01	Data should be ingested through an API endpoint
SS-CR-02	All activities performed within the tool should be logged for auditability
SS-CR-03	Data should be able to be ingested concurrently
SS-CR-04	Data storage should be consistent with data received by the system
SS-CR-05	PDE data must be hashed for integrity verification

SS-CR-06	Metadata must be sanitized for security purposes
SS-CR-07	PDE and sensitive metadata must be encrypted using strong encryption
SS-CR-08	The hash digest and metadata should be viewable by an investigator
SS-CR-09	The tool must provide digests of the encrypted PDE to ensure its forensic soundness
SS-CR-10	Each PDE should be distinguishable from another anonymously
SS-CR-11	Ingested data collected should be displayed
SS-CR-12	Ingested data must be validated for correctness
SS-CR-13	Authorization and authentication must be implemented fully for access control
SS-CR-14	An investigator should be able to securely download the PDE

Table 2. Secure Storage Optional Requirements (SS-OR)

Label	Description
SS-OR-01	All metadata data should be encrypted for anonymity
SS-OR-02	PDE can only be decrypted on a successful session and 2FA validation
SS-OR-03	All stored PDE for specific user permissions should be viewable
SS-OR-04	Malicious data should not be ingested and stored as PDE
SS-OR-05	PDE should not be subject to URL manipulation

B. SecureRS SYSTEM IMPLEMENTATION

Now that the requirements have been defined, the tool was implemented using a modular approach and applying agile principles. The programming language that was chosen to implement this tool was Python, due to its flexibility and built-in frameworks and libraries. In order to make a web platform and allow easy management, Django web framework [36] was chosen. The tool uses Django REST framework [37] as it provides a mechanism for applying RESTful API functionality fairly easily. This framework furthermore provides authentication based on a secure API key, which is created from the admin panel on the system, allowing easy management and revoking of keys. Each key is unique and serves as an authentication mechanism for making an HTTP POST request to the API endpoint. The security sanitization process followed uses Django's default security middleware as well as custom sanitization middleware to remove special characters and tags from the metadata. The different middlewares used include: SecurityMiddleware, SessionMiddleware, CsrfViewMiddleware, AuthenticationMiddleware, XFrameOptionsMiddleware, OTPMiddleware, SessionSecurityMiddleware. The tool also made use of encrypted fields in Django models to further protect the sensitive metadata. This was done to prevent the misuse of data due to unauthorized access or misconduct.

To secure the PDE file, a Django-encrypted file field was chosen, which uses the Fernet encryption scheme [38]. The latter is a symmetric key algorithm that makes sure that the encrypted message cannot be manipulated, brute-forced, or read without a password key. This key is URL safe encoded with base64 so that any reserved, unprintable, or non-ASCII characters are replaced. It ensures that no errors occur when handling the keys that an attacker could potentially exploit. Fernet also makes use of advanced encryption standard (AES) 128-bit cipher block chaining (CBC) mode and public-key cryptographic standards number 7 (PKCS7) padding. This means that the cipher is in multiples of 128-bits, with PKCS7 padding to fill the remaining bits. The password key makes use of a hash-based message authentication code (HMAC). The HMAC serves a dual purpose and simultaneously verifies the integrity and authenticity of a message. This is done to ensure better security, as HMAC was used in conjunction with a simple hashing algorithm (SHA) of 256 bits (SHA256) [38]. All symmetric encryption keys are on the system itself, either as a setting in the Django configuration or managed by the Django framework itself.

Portability was one of the contributing factors for making use of Docker [39] (a containerized approach to hosting services). Using Docker makes the system easily scalable as well as platform independent, and it provides load balancing. The high-level flow chart showing the lifecycle of SecureRS is shown in Figure 4. The lifecycle starts with ensuring that the system is installed successfully, and subsequently initializes the system. The setup does not require much besides creating a superuser (a feature of the Django framework) and it provides admin functionality such as creating users, setting access roles, creating API keys, etc.

Once the system has been initialized, the next step is account creation, which involves two-factor authentication (2FA), and the creation of API keys. From a design perspective, it was decided that (for more security and traceability) only an admin user can create users and API keys. The 2FA system catered for email, SMS, and YubiKey [40]. 2FA is required for logging in and also to ensure safe download of a PDE file. Token generators make use of the time-based one-time pin (TOTP) algorithm [41] that generates 6-8 unique digits based on the current time and some secret key that is added when the account is registered. By design, these tokens are regenerated every 30 seconds to prevent attackers from brute-forcing the token or launching phishing attacks. Backup tokens are also enabled if devices used for TOTP are not available. These backup tokens can only be used once, thereby allowing recovery and security. User credentials are stored using Django's default password field, which uses PBKDF2 with strong

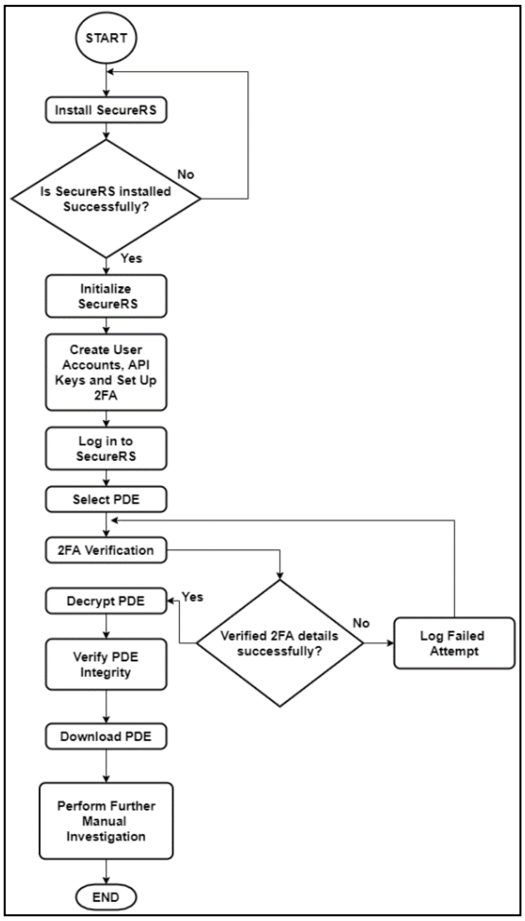


Figure 4. **High-level Lifecycle of SecureRS**

SHA 256-bit hashing and a random salt [36]. This password stretching mechanism is recommended by NIST [35]. When an investigator selects a PDE file to be investigated further, several checks occur. Firstly, the session is checked to see if it is still active and if the logged-in user has the required permissions. This is achieved by verifying that the inactivity time of the user has not passed the threshold and that 2FA is enabled. A user is warned after three minutes of inactivity and consequently logged out after ten minutes of inactivity. These thresholds are configurable in the settings. After the session and 2FA process have been successfully validated, the PDE will be decrypted by the system and available for the investigator to download for further manual investigation to corroborate findings. The implementation of SecureRS platform can be found at: https://github.com/AvinashSingh786/SecureRS

C. SECURE STORAGE VALIDATION

This section details the testing of the tool in terms of its implementation correctness and determines if the tool has met the requirements defined in Section A. This complies with the NIST validation cycle and is structured as follows: secure storage core test assertions (SS-CA) (see Table 3), secure storage test cases (SS-TC) (see Table 4), and the secure storage compliance matrix (SSCM) (see Table 5).

Table 3. Secure Storage Core Test Assertions (SS-CA)

Label	Description
SS-CA-01	All PDE files should be encrypted. **Justification:** To preserve sensitive information and confidentiality and to prevent unauthorized access to the PDE.
SS-CA-02	Hash digests of the PDE should occur before and after PDE encryption. **Justification:** To maintain the integrity of the PDE as well as to ensure no errors occur or modifications are made to the PDE itself.
SS-CA-03	Audit logging of all activity is maintained. **Justification:** This ensures that the chain of custody is maintained and that the process followed is reliable and verifiable.
SS-CA-04	Metadata sanitization of ingested data. **Justification:** To ensure good security practices and prevent potential system attacks. This reduces the attack vectors from injection attacks like XSS, SQL injection, and parsing attacks.

Table 4. Secure Storage Test Cases (SS-TC)

Label	Description
SS-TC-01	Make a POST request with correct and relevant data, to verify that the PDE and metadata were successfully added.
SS-TC-02	Make a malicious POST request with XSS and SQL injection payloads and see if they are sanitized and averted.
SS-TC-03	Make a POST request with an incorrect API token and check if the POST request is denied.
SS-TC-04	Send multiple instances of invalid data to test the validation process.
SS-TC-05	Manually hash the PDE before encryption and test if the hash digests match.
SS-TC-06	Download the PDE and see if it matches the original database hash digest.
SS-TC-07	Verify the timestamp of the database and the file timestamp.
SS-TC-08	Perform URL manipulation to attempt to download PDE.
SS-TC-09	Try to download PDE without 2FA authentication enabled.
SS-TC-10	Verify if 2FA works as expected.

A compliance matrix simply states the requirements, the test case(s) that tested the specific requirement, and the result, which is a core test assertion or a manual check. For example, if a core test assertion was met, that test assertion is specified in the result column. However, if a manual check was performed, it is indicated with '--check--', indicating that the check result is compliant. The compliance matrix determines if the tool met the requirements and is compliant. The compliance matrix for secure storage is presented in Table 5. The compliance matrix confirms that the results from the test assertions have been fulfilled, therefore implying that all the requirements defined have been met and are successfully tested and compliant.

Table 5. Secure Storage Compliance Matrix

#	Requirement	Test case	Result
1	SS-CR-01	SS-TC-01	--check--
2	SS-CR-02	SS-TC-02	SS-CA-01
3	SS-CR-03	SS-TC-01	--check--
4	SS-CR-04	SS-TC-02, SS-TC-08, SS-TC-04	SS-CA-01, SS-CA-02, SS-CA-03, SS-CA-04
5	SS-CR-05	SS-TC-05	SS-CA-02
6	SS-CR-06	SS-TC-02	SS-CA-04
7	SS-CR-07	--check--	--check--
8	SS-CR-08	SS-TC-01	SS-CA-03
9	SS-CR-09	SS-TC-02	SS-CA-01
10	SS-CR-10	SS-TC-05, SS-TC-06	SS-CA-03, --check--
11	SS-CR-11	SS-TC-01	--check--
12	SS-CR-12	SS-TC-02, SS-TC-03, SS-TC-07, SS-TC-08	SS-CA-03, SS-CA-04
13	SS-CR-13	SS-TC-09	--check--
14	SS-CR-14	SS-TC-01	--check--
15	SS-OR-01	SS-TC-05	--check--
16	SS-OR-02	SS-TC-06	SS-CA-03
17	SS-OR-03	--check--	--check--
18	SS-OR-04	SS-TC-03	SS-CA-02
19	SS-OR-05	SS-TC-08	--check--

Now that the tool has been validated and satisfies the NIST CFTT [34], the next phase is to evaluate the tool to determine the performance of the system and its model.

V. SecureRS PERFORMANCE EVALUATION

Given that the prototype system has been validated through the NIST CFTT [34], it is important to gauge the performance of the system. To determine the performance of the system, several factors were considered – the speed of data ingestion; the amount of data ingested; processor and memory utilization; the time until the data was stored. Table 6 shows the system specification that was used to benchmark the application of SecureRS. Organizations typically would run the system on the same network; therefore, to rule out network speeds and latency, the system was tested in ideal circumstances where the data ingested was sourced from the same host, i.e., 'localhost'.

Table 6. Benchmarking System Specifications

Item	Description	Specification
CPU	Intel Core i7	I7-7700HQ @ 2.8GHz
RAM	DDR4	24 GB @ 2400MHz
DISK	NVME SSD	1 TB @ Read: 3500 MB/s, Write: 3000 MB/s
OS	Windows 10	

It is also important to determine a baseline of the SecureRS systems memory and processor utilization so that when data is being ingested, the overall performance difference can be determined. Table 7 shows the idle baseline values for the system. From this table, an approximate value of 36 MB of memory utilization and about 1% of processor utilization are used, showing that when no data is being ingested and stored, the system does not utilize many resources.

Table 7. Baseline Processor and Memory Utilization of SecureRS

Item	Baseline values
Idle memory consumption	~ 36 MB
Idle processor utilization	~ 1%

The testing phase consisted of a 1 GB PDE and 100 MB payload that contained methodically generated ASCII data. In DFR, the size of a PDE is context-dependent and relies on the kind of data that is stored. It is quite difficult to get an accurate representation of the maximum size of a PDE payload. However, this research is aimed at DFR, so large PDE files would be extremely rare. To that end, tests were performed on the perceived worst-case and best-case size of a PDE to determine the effects on performance. For this study, the worst case chosen was a large payload of 1 GB and the average case was 100 MB. These values were chosen based on some existing DFR literature [42]–[45]. To make the performance evaluation as comprehensive as possible, the system was tested under various circumstances, namely single, multiple, and concurrent HTTP requests. In the case of single requests, only one HTTP POST request was made, and the performance indicators were observed. Multiple HTTP requests were sent synchronously, meaning that after one request was sent, another was initiated. To get an average, a set of 10 requests was made. The reason for testing synchronously was to determine the implications of the hashing process conducted by the system and to see if it would be able to handle the load without using many resources. The final evaluation was based on sending concurrent requests to the system to see how its performance would be affected and to show the robustness of the system. Table 8 shows the performance of a single request with a 1 GB PDE payload. The results in parentheses show the performance at the time of hashing and encryption. From these results, an average of 3.6% processor utilization was used while 19.6% was used during hashing and encryption. We observed a higher memory utilization during hashing and encryption, apparently because parts of the file must be read into memory before it can be encrypted and hashed. Overall, for the worst case of a 1 GB PDE, a total time of 36.76s was observed to ingest, validate, secure, and store. Where an investigator would perform the process manually, it would take roughly 2m 11s, based on one manual attempt conducted by the authors.

When comparing Table 8 and Table 9, there is not much difference in the performance. This was expected since the requests were sent synchronously. An average time of 36.21s was observed from the time the PDE was sent to the storage engine until the time it was successfully stored following the forensic assurance processes. Table 10 shows the performance for concurrent requests, and a slight decrease in ingestion speed and an increase in CPU usage could be observed. This was expected, as requests were performed in parallel.

Table 8. Performance of 1 GB PDE with Single Request

Item (1 GB PDE)	**Single request (during encryption)**
Speed of data ingestion	102 MB/s
Amount of data ingested	1.33 GB
Processor utilization	3.6% (19.6%)
Memory utilization	30.2 MB (4.3 GB)
Time	36.76s (16.3s)

Table 9. Performance of 1GB PDE with Multiple Requests

Item (1 GB PDE)	**Multiple single requests {10} (during encryption)**
Average speed of data ingestion	110 MB/s
Total amount of data ingested	14.6 GB
Average. processor utilization	3.4% (16.2%)
Average memory utilization	36.6 MB (5.2 GB)
Average time per request	36.21s (18.9s)

Table 10. Performance of 1GB PDE with Concurrent Requests

Item (1 GB PDE)	**Concurrent request {10} (during encryption)**
Average speed of data ingestion	108 MB/s
Total amount of data ingested	13.3 GB
Average processor utilization	16.4% (46.5%)
Average memory utilization	33.5 MB (6.6 GB)
Total average time	1m 34s (22.3s)

Table 11 to Table 13 show that where a smaller PDE was used, a corresponding insignificant difference in performance was observed. This implies that the system can still perform well under high load without a significant time utilization; however, as expected, it does consume more resources. The bottleneck occurs when hashing and encryption are performed, since this is a computationally expensive task. Even though during the concurrent requests there was more processor and memory utilization, the system still performed well given the process each PDE had to undergo. Results from Table 8 to Table 13 clearly show that SecureRS can still ingest data relatively well and is able to handle the load without much resource usage. Moreover, forensic soundness is ensured through the defined processes.

Table 11. Performance of 100 MB PDE with Single Request

Item (100 MB PDE)	**Single request (during encryption)**
Speed of data ingestion	40.3 MB/s
Amount of data ingested	133 MB
Processor utilization	1.6% (10.6%)
Memory utilization	30.2 MB (144 MB)
Time	5.02 (4.4s)

Table 12. Performance of 100 MB PDE with Multiple Requests

Item (100 MB PDE)	**Multiple single requests {10} (during encryption)**
Average speed of data ingestion	50.1 MB/s
Total amount of data ingested	1.33 GB
Average processor utilization	4.4% (13.2%)
Average memory utilization	43.6 MB (117 MB)
Average time per request	7.21s (6.1s)

Table 13. Performance of 100 MB PDE with Concurrent Requests

Item (100 MB PDE)	**Concurrent request {10} (during encryption)**
Average speed of data ingestion	99 MB/s
Total amount of data ingested	1.3 GB
Average processor utilization	14.4% (43.5%)
Average memory utilization	43.5 MB (2.1 GB)
Total average time	11.3s (3.4s)

To further illustrate the effectiveness of SecureRS, a graphical depiction of Table 8 to 13 is illustrated in Figure 5. From this figure, the observed speed of data ingestion remained relatively consistent with relation to the PDE size and operation. The memory consumption remained somewhat consistent over the tests excluding the encryption state. The concurrent processor utilization for both the 1GB and 100MB PDE remained in a same range between 14-16% whereas the time was significantly better with the larger PDE. This is on the assumption that 1GB test is 10 times that of the 100MB test. This therefore suggests that if the 100MB concurrent test took 11.3s, the 1GB one would be estimated around 113s, and the actual value was 94s. This therefore demonstrates the efficacy of the SecureRS system. A similar deduction can be made when looking at the speed of data ingestion for the concurrent tests factoring in the network limitations.

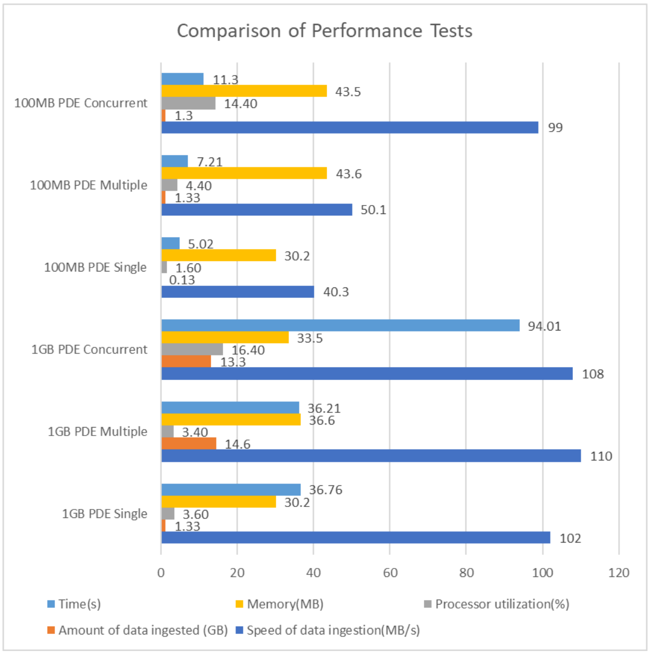


Figure 5. **Comparison of performance tests**

A test was conducted utilizing a larger PDE of 10 GB and the overall time taken was 2m 24s. The average CPU was 12%, with the total amount of data stored being 13.3 GB. Although this system was developed with small artifacts in mind, it can cater for file sizes as large as the system's memory, due to the limitations of the Fernet encryption python library. This limitation can easily be addressed by file-streaming the information instead; however, this falls outside of the scope of DFR and the research at hand. The next section discusses what the proposed model achieved and how this can aid the forensic investigation lifecycle.

VI. DISCUSSION AND CASE SCENARIO

In traditional digital forensic processes, investigators are often expected to follow the correct procedure and protocol. However, human error can occur. For example, several litigation proceedings have resulted in exculpatory outcomes due to digital evidence mishandling [46], [47]. However, automating and providing a storage engine with forensics and security in place, significantly aids an investigator. For instance, the investigator does not need to be concerned about safely securing artefacts or data that contains sensitive information. Furthermore, the threat of privacy concerns and integrity violation, which has been associated with poor digital evidence handling, can be reduced when human elements are restricted. Therefore, the forensic storage process developed in this study can be defined as the potential steps towards addressing these challenges. This system works well for digital forensic readiness whereby potential digital evidence is collected on the fly.

A case scenario of the use of the SecureRS model is to be a plugin into a collection model. For example, the previous work by the authors involved the collection of forensic artefacts from a ransomware attack using digital forensic readiness [48]. Such previous work involved collecting small-sized PDE files and storing them for further investigation. While the collection on its own is a major contribution, the authors did not ensure the extra measures to protect the PDE and ensure that it was forensically sound and admissible in a court of law. The SecureRS model solves this problem and helps other research within DFR to the extent that developed frameworks or systems do not need to be concerned about the storage and preservation of the potential digital evidence collected. However, integrating this peculiar notion of secure storage for digital investigation was quoted as potential future research. SecureRS can aid in ensuring the integrity of the collected PDE. Furthermore, the model developed in [43] asserts that the use of security standards like encryption and hashing can be used to achieve confidentiality and integrity. Based on the performance evaluation, the model developed in [43] has a low impact on a system whilst providing a core and essential service. Extending this previous study, SecureRS provides a feasibility and proof-of-concept implementation of automated evidence storage. By integrating a reliable forensic process and practice, SecureRS provides a platform for developing a limited human interaction with potential digital evidence.

VII. SECURITY ANALYSIS

To further evaluate the developed SecureRS tool and model, a threat modelling and security analysis process was followed (see Table 14). In this model, several security features were used to protect and maintain the integrity and forensic soundness of the data stored. This was achieved by using the CIA triad and several security requirements. For instance, threats due to filename change or deletion was addressed in SecureRS using randomization of file name, and permission-based access control such that only permitted action (by the authorized entity) is allowed. Furthermore, the log of such an action is provided for each instance. This further addressed the need for accountability (system audit process) within the system. The SecureRS thus provide a forensic platform that can mitigate such a threat. Similarly, to ensure confidentiality and prevent the potential of sensitive information leakage, SecureRS leverages an encryption algorithm with stronger immunity.

Table 14. Threat-Solution model using the CIA triad

Threat	Solution	Method
Access to sensitive information	Implementing encryption (Confidentiality and Authorization)	Fernet encryption
Attacker changes filenames or deleting files	Applying permissions (Integrity, Availability, and Authorization)	Random file names and read-only permissions
Attacker changes contents of PDE data in memory	Implementing hashing (Confidentiality, Integrity, and Authorization)	MD5 hashing is used to compare source MD5 of the PDE file and received PDE file. Also, OS memory protection is used
Attacker finds credentials and can login	Implementing 2FA (Confidentiality, Authenticity, and Authorization)	TOTP is used for PDE downloading
Attacker gains physical access to a computer after investigator goes outside	Implementing session timeouts (Confidentiality, Availability, and Authenticity)	Customizable activity and session timers are used to prevent unauthorized access
XSS and injection attacks	Implementing data sanitization (Confidentiality, Integrity, and Availability)	Django middleware and HSTS + secure cookies
Investigator denying PDE download	Implementing logs (Confidentiality and Nonrepudiation)	Audit logs and emails are used to verify activity and download PDE
Interception of PDE in transit	Implementing HTTPS and API Keys (Confidentiality, Integrity, Authenticity, and Nonrepudiation)	Using SSL and API keys that are unique reduces the risk of data exposure.

VIII. FUTURE WORK

Future work will consider extending the platform to provide lossless compression [49] and storage optimization, developing novel methods for data ingestion, and well as develop novel methods for PDE relevance categorization. Also, potential approaches towards cloud-based evidence storage in a readiness manner will be further considered as well as extending to other sub-domains of digital forensics. As asserted in recent studies within the forensic community [50], [51], the development of a generic platform of SecureRS can be a potential solution to the lack of standardized evidence representation, as well as unified metrics towards evidence reliability evaluation/testing,

IX. CONCLUSION

In conclusion, this paper developed a model and a platform to secure Potential Digital Evidence (PDE) and to ensure the forensic soundness of the stored PDE. The platform was evaluated and shown to render good performance, despite having to go through all the forensic processes defined by the proposed model (SecureRS). Having a process in place to secure evidence can help prevent unauthorized access and comply with regulations and privacy policies, due to the nature of the data being stored. Having this model in place also helps to verify and validate the stored PDE and make it admissible in a court of law. By leveraging encryption and hashing, the SecureRS model makes good use of current security standards and therefore will aid forensic investigation in general. The model also helps to detect evidence tampering. This paper suggests a method of ensuring forensically sound digital evidence for DFR as well as for digital forensics processes in general. So far, this aspect of forensics investigation has been widely overlooked and it was often considered to be the sole responsibility of the forensic investigator. The focus and scope of this study involved smaller artefacts for performance evaluation. With SecureRS an investigator does not need to be concerned about verification and authenticity of evidence when performing a digital investigation. The SecureRS platform furthermore acts as a backup of evidence that is securely and safely stored.

REFERENCES

[1] D. Gonzalez and T. Hayajneh, “Detection and prevention of crypto-ransomware,” in *2017 IEEE 8th Annual Ubiquitous Computing, Electronics and Mobile Communication Conference, UEMCON 2017*, 2018, vol. 2018-January, doi: 10.1109/UEMCON.2017.8249052.

[2] R. Von Solms and J. Van Niekerk, “From information security to cyber security,” *Computers and Security*, 2013, doi: 10.1016/j.cose.2013.04.004.

[3] H. de Bruijn and M. Janssen, “Building Cybersecurity Awareness: The need for evidence-based framing strategies,” *Government Information*

Quarterly, vol. 34, no. 1, pp. 1–7, 2017, doi: 10.1016/j.giq.2017.02.007.

[4] A. M. Maigida, S. M. Abdulhamid, M. Olalere, J. K. Alhassan, H. Chiroma, and E. G. Dada, “Systematic literature review and metadata analysis of ransomware attacks and detection mechanisms,” *Journal of Reliable Intelligent Environments*, vol. 5, no. 2, pp. 67–89, 2019, doi: 10.1007/s40860-019-00080-3.

[5] E. Casey, “Error, Uncertainty, and Loss in Digital Evidence,” *International Journal of Digital Evidence*, 2002.

[6] C. Altheide and H. Carvey, “Windows Systems and Artifacts,” in *Digital Forensics with Open Source Tools*, 2011.

[7] N. C. Rowe, “Associating Drives Based on Their Artifact and Metadata Distributions,” in *Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, LNICST*, 2019, vol. 259, pp. 165–182, doi: 10.1007/978-3-030-05487-8_9.

[8] D. Lillis, B. Becker, T. O’Sullivan, and M. Scanlon, “Current Challenges and Future Research Areas for Digital Forensic Investigation,” 2016, [Online]. Available: http://arxiv.org/abs/1604.03850.

[9] J. Dokko and M. Shin, “A Digital Forensic Investigation and Verification Model for Industrial Espionage,” in *Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, LNICST*, vol. 259, 2019, pp. 128–146.

[10] A. Shabtai, Y. Elovici, and L. Rokach, “Introduction to Information Security,” in *SpringerBriefs in Computer Science*, 2012.

[11] R. Bhanot and R. Hans, “A review and comparative analysis of various encryption algorithms,” *International Journal of Security and its Applications*, 2015, doi: 10.14257/ijsia.2015.9.4.27.

[12] M. Kohn, M. S. Olivier, and J. H. P. Eloff, “Framework for a Digital Forensic Investigation.,” *Communications*, 2006.

[13] J. Tan, “Forensic readiness,” *Cambridge*, pp. 1–23, 2001, doi: 10.1.1.644.9645.

[14] ISO 27043, “INTERNATIONAL STANDARD ISO / IEC 27043: Information technology — Security techniques — Incident investigation principles and processes,” vol. 2015, 2015.

[15] S. M. Makura, H. S. Venter, R. A. Ikuesan, V. R. Kebande, and N. M. Karie, “Proactive Forensics: Keystroke Logging from the Cloud as Potential Digital Evidence for Forensic Readiness Purposes,” *2020 IEEE International Conference on Informatics, IoT, and Enabling Technologies, ICIoT 2020*, pp. 200–205, 2020, doi: 10.1109/ICIoT48696.2020.9089494.

[16] S. Vömel and J. Stüttgen, “An evaluation platform for forensic memory acquisition software,” in *Digital Investigation*, 2013, vol. 10, no. SUPPL., pp. S30–S40, doi: 10.1016/j.diin.2013.06.004.

[17] M. Gruhn and F. C. Freiling, “Evaluating atomicity, and integrity of correct memory acquisition methods,” *Digital Investigation*, vol. 16, pp. S1–S10, 2016, doi: 10.1016/j.diin.2016.01.003.

[18] M. Harran, W. Farrelly, and K. Curran, “A method for verifying integrity & authenticating digital media,” *Applied Computing and Informatics*, 2018, doi: 10.1016/j.aci.2017.05.006.

[19] R. Sears, C. Van Ingen, and J. Gray, “To BLOB or Not To BLOB: Large Object Storage in a Database or a Filesystem?,” *Microsoft Research*, pp. 1–11, Jan. 2007, [Online]. Available: http://arxiv.org/abs/cs/0701168.

[20] S. Pandey and M. Farik, “Best Symmetric Key Encryption - A Review,” *International Journal of Scientific & Technology Research*, vol. 6, no. 6, pp. 310–312, 2017.

[21] D. S. Abd Elminaam, H. M. A. Kader, and M. M. Hadhoud, “Evaluating the performance of symmetric encryption algorithms,” *International Journal of Network Security*, vol. 10, no. 3, pp. 213–219, 2010, doi: 10.6633/IJNS.201005.10(3).06.

[22] J. Van der Walt and R. Luke, “The storage of forensic evidence at the Forensic Science Laboratory in Pretoria, South Africa,” *Journal of Transport and Supply Chain Management*, vol. 5, no. 1, pp. 202–220, 2011, doi: 10.4102/jtscm.v5i1.74.

[23] A. Valjarevic, H. Venter, and R. Petrovic, “ISO/IEC 27043:2015 - Role and application,” *24th Telecommunications Forum, TELFOR 2016*, pp. 1–4, 2017, doi: 10.1109/TELFOR.2016.7818718.

[24] X. Du, N.-A. Le-Khac, and M. Scanlon, “Evaluation of Digital Forensic Process Models with Respect to Digital Forensics as a Service,” 2017, [Online]. Available: http://arxiv.org/abs/1708.01730.

[25] D. R. Rani and G. Geethakumari, “Secure data transmission and detection of anti-forensic attacks in cloud environment using MECC and DLMNN,” *Computer Communications*, vol. 150, no. November 2019, pp. 799–810, 2020, doi: 10.1016/j.comcom.2019.11.048.

[26] P. M. Trenwith and H. S. Venter, “Digital forensic readiness in the cloud,” in *2013 Information Security for South Africa*, Aug. 2013, pp. 1–5, doi: 10.1109/ISSA.2013.6641055.

[27] B. K. S. P. K. Raju and G. Geethakumari, “An advanced forensic readiness model for the cloud environment,” *Proceeding - IEEE International Conference on Computing, Communication and Automation, ICCCA 2016*, pp. 765–771, 2017, doi: 10.1109/CCAA.2016.7813819.

[28] V. R. Kebande and H. S. Venter, “A Cloud

Forensic Readiness Model Using a Botnet as a Service," in *The International Conference on Digital Security and Forensics (DigitalSec2014)*, 2014, pp. 23–32, doi: 10.13140/2.1.4880.2249.

[29] A. Singh, "A Digital Forensic Readiness Approach for Ransomware Forensics," *UP Space*, no. August, 2019, [Online]. Available: https://repository.up.ac.za/bitstream/handle/2263/75610/Singh_Digital_2019.pdf.

[30] M. L. Das and N. Samdaria, "On the security of SSL/TLS-enabled applications," *Applied Computing and Informatics*, 2014, doi: 10.1016/j.aci.2014.02.001.

[31] S. Omeleze and H. S. Venter, "Proof of concept of the online neighbourhood watch system," in *Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, LNICST*, 2016, vol. 171, pp. 78–93, doi: 10.1007/978-3-319-43696-8_9.

[32] L. Chi and X. Zhu, "Hashing Techniques: A Survey and Taxonomy," *ACM Computing Surveys (CSUR)*, 2017, doi: 10.1145/3047307.

[33] E. M. Schön, J. Thomaschewski, and M. J. Escalona, "Agile Requirements Engineering: A systematic literature review," *Computer Standards and Interfaces*, vol. 49, pp. 79–91, 2017, doi: 10.1016/j.csi.2016.08.011.

[34] NIST, "Computer Forensics Tool Testing Program," 2014. http://www.cftt.nist.gov/disk_imaging.htm (accessed Oct. 12, 2017).

[35] NIST, "NIST," 2019. https://www.nist.gov/ (accessed Oct. 12, 2019).

[36] DjangoProject, "Django Auth." https://docs.djangoproject.com/en/2.2/topics/auth/passwords/ (accessed Oct. 04, 2019).

[37] "Django REST framework," 2019. http://www.django-rest-framework.org/#django-rest-framework (accessed Feb. 14, 2019).

[38] P. Dijesh, S. Babu, and Y. Vijayalakshmi, "Enhancement of e-commerce security through asymmetric key algorithm," *Computer Communications*, vol. 153, pp. 125–134, Mar. 2020, doi: 10.1016/j.comcom.2020.01.033.

[39] T. Combe, A. Martin, and R. Di Pietro, "To Docker or Not to Docker: A Security Perspective," *IEEE Cloud Computing*, 2016, doi: 10.1109/MCC.2016.100.

[40] B. Schneier, "Two-factor authentication: Too little, too late," *Communications of the ACM*, vol. 48, no. 4, p. 136, 2005, doi: 10.1145/1053291.1053327.

[41] J. R. D. M'Raihi, S. Machani, M. Pei, "TOTP: Time-Based One-Time Password Algorithm," *Journal of Chemical Information and Modeling*, 2013, doi: 10.1017/CBO9781107415324.004.

[42] S. Philomin, A. Singh, A. Ikuesan, and H. Venter, "Digital forensic readiness framework for smart homes," 2020, doi: 10.34190/ICCWS.20.047.

[43] A. Singh, A. Ikuesan, and H. Venter, "A context-aware trigger mechanism for ransomware forensics," in *14th International Conference on Cyber Warfare and Security, ICCWS 2019*, 2019, pp. 629–638.

[44] M. Lagrasse, A. Singh, H. Munkhondya, A. Ikuesan, and H. Venter, "Digital forensic readiness framework for software-defined networks using a trigger-based collection mechanism," in *Proceedings of the 15th International Conference on Cyber Warfare and Security, ICCWS 2020*, 2020, pp. 296–305, doi: 10.34190/ICCWS.20.045.

[45] H. Munkhondya, A. Ikuesan, and H. Venter, "Digital Forensic Readiness Approach for Potential Evidence Preservation in Software-Defined Networks," 2015.

[46] S. E. Goodison, R. C. Davis, and B. A. Jackson, "Digital evidence and the U.S. criminal justice system," *Priority Criminal Justice Needs Initiative*, pp. 1–32, 2015.

[47] H. Arshad, A. Bin Jantan, and O. I. Abiodun, "Digital forensics: Review of issues in scientific validation of digital evidence," *Journal of Information Processing Systems*, vol. 14, no. 2, pp. 346–376, 2018, doi: 10.3745/JIPS.03.0095.

[48] A. Singh, A. R. Ikuesan, and H. S. Venter, "Digital Forensic Readiness Framework for Ransomware Investigation," *Digital Forensics and Cyber Crime*, vol. 259, pp. 91–105, 2019, doi: 10.1007/978-3-030-05487-8_5.

[49] A. Moffat, "Lossless Compression," *The Computer Journal*, 1997, doi: 10.1093/comjnl/40.2_and_3.65.

[50] A. Al-Dhaqm, S. Razak, K. Siddique, R. A. Ikuesan, and V. R. Kebande, "Towards the Development of an Integrated Incident Response Model for Database Forensic Investigation Field," *IEEE Access*, p. 1, 2020, doi: 10.1109/ACCESS.2020.3008696.

[51] A. Al-Dhaqm, S. A. Razak, R. A. Ikuesan, V. R. Kebande, and K. Siddique, "A Review of Mobile Forensic Investigation Process Models," *IEEE Access*, vol. 8, pp. 173359–173375, 2020, doi: 10.1109/access.2020.3014615.

AVINASH SINGH is a researcher in the Digital Forensic space, focusing on Ransomware and Malware Forensics as well as on Digital Forensic Readiness. He obtained both the BSc Hons and MSc in Computer Science with distinction from the University of Pretoria. He is currently a lecturer in the Department of Computer Science and is pursuing a PhD. Avinash participated in numerous international conferences and has published in various journals. He is a member of the review committee for the Information Security South Africa (ISSA) conference, a member of the Golden Key Society, IITPSA, and the Digital Forensic Science (DigiForS) Research Group. He also heads the recently established Intelligent Cyber Forensic Lab (ICFL) at the University of Pretoria.

RICHARD ADEYEMI IKUESAN is an active researcher who currently pioneers a digital policing and forensic project for developing nations, using Nigeria and South Africa as a hub for West Africa and Southern Africa respectively. He obtained the MSc and PhD degrees in Computer Science with distinction from the Universiti Teknologi Malaysia and is currently an Assistant Professor in the Cyber Security section of the IT department at the Community College of Qatar.

HEIN VENTER has established an international research reputation in cyber security and cyber forensics. Over the past 13 years, he has focused mainly on cyber forensics research. Hein is the research group leader for the Digital Forensic Science (DigiForS) research group at the University of Pretoria where he collectively supervises more than 40 Computer Science postgraduate students. He authored and co-authored more than 260 publications. He is also general chair of the Information Security for South Africa (ISSA) conference. Hein recently served on a panel that was tasked by the South African Department of Science and Technology (DST) to come up with a national cyber security research agenda. The main topics identified for this research agenda include cyber security and digital forensics.