UDC 004.02
DOI: 10.15587/2706-5448.2025.329386

Maryna Larchenko

DEVELOPMENT OF A SECURE STORAGE ARCHITECTURE FOR DIGITAL EVIDENCE

The object of the study is the process of generating, transmitting, and storing memory dumps within digital forensics. The problem being addressed is the insufficient level of security of existing methods of transmitting and storing digital evidence, which can lead to their compromise, loss of authenticity, and inadmissibility in court proceedings.

As a result of the conducted research, an architecture for secure storage of digital evidence was developed, providing protection at the stages of acquisition, transportation, storage, and further analysis of memory dumps. A cross-platform Python script for automated memory dump acquisition was proposed, as well as a mechanism for secure transportation of evidence using cryptographic protection through the SCP protocol and authentication. The effectiveness of the combined use of SSH encryption, creation of file system containers in "read-only" mode, mandatory logging of all actions with digital evidence, and an integrated hash-checking mechanism for data integrity verification was demonstrated.

The effectiveness of the proposed approach was assessed based on process modeling in a test environment. In particular, the collected memory dumps were transferred using a custom Python script using a "safe corridor" from the Kali Linux virtual machine to the Caine virtual machine to the created container in "read-only" mode. The integrity of the files after transportation and storage was checked using a hash sum comparison.

A distinctive feature of the proposed model is a comprehensive approach to digital evidence protection, combining technical and organizational measures to ensure the authenticity and integrity of data. This allows solving the problem of compromising digital evidence and guarantees its judicial admissibility. The results obtained are explained by the implementation of cryptographic methods and compliance with digital forensics standards.

The proposed methodology can be used in the practice of law enforcement agencies, forensic experts, as well as in the development of national standards for the preservation of digital evidence. The storage model complies with international security standards and can be adapted to the specific requirements of judicial proceedings in Ukraine.

Keywords: *digital forensics, memory dumps, cryptographic protection, cross-platform Python script, file system containers, "read-only" mode.*

Received: 07.02.2025
Received in revised form: 03.04.2025
Accepted: 26.04.2025
Published: 13.05.2025

How to cite
Larchenko, M. (2025). Development of a secure storage architecture for digital evidence. Technology Audit and Production Reserves, 3 (2 (83)), 33–43. https://doi.org/10.15587/2706-5448.2025.329386

1. Introduction

In today's environment, digital forensics plays a key role in the investigation of crimes involving the use of computer systems. One of the most valuable sources of information in this process is RAM dumps (casts), which contain traces of user activity that may be lost after the device is turned off. Therefore, an important task is to ensure the reliable creation of memory dumps and their secure transfer to specialized storages for further analysis.

The technical complexity of this process is due to the need to guarantee the integrity of data during transmission over secure communication channels. Creating memory dumps involves the use of specialized tools, such as LiME, Volatility or Rekall, which allow to take snapshots of memory in real time. However, once the dump is obtained, the issue of its safe transportation arises. Using unprotected channels or uncontrolled environments can lead to data modification or compromise. Therefore, the introduction of the concept of a "secure corridor" for the transmission of memory dumps is a necessary measure to preserve the authenticity of digital evidence.

The proposed study considers the technical aspects of building such the "secure corridor", including encryption methods, integrity checking mechanisms and methods of data source authentication. Existing approaches are analyzed and an optimized solution is proposed to ensure the secure transport of digital evidence to an isolated storage facility (read-only) without the risk of unauthorized modification or loss.

The digital evidence obtained in this way is of significant value not only in criminal proceedings, but also in civil, administrative and arbitration cases. For example, in the area of corporate security, the analysis of memory dumps can be used to investigate industrial espionage or confidential information leakage. In litigation related to labor disputes or copyright infringement, digital evidence can confirm the fact of unauthorized access or manipulation of data. In international investigations, these methods are becoming a tool for proving cyberattacks and illegal activities in cyberspace.

Currently, all available solutions for the secure transfer of memory dumps are commercial, which makes it difficult for them to be widely adopted by government and law enforcement agencies. The lack of free tools makes it difficult for researchers and experts to access effective

ISSN-L 2664-9969; E-ISSN 2706-5448

methods of preserving digital evidence. This opens up prospects for the development of solutions that can become a standard for digital forensics and help improve security in the investigation of digital crimes.

Thus, research on the architecture of secure storage for digital evidence and methods of its secure transfer is relevant for the development of modern digital forensics and the improvement of the effectiveness of the use of digital evidence in law enforcement practice.

In modern digital forensics, the analysis of RAM dumps is a critical step in the detection and investigation of criminal activity involving computer systems. Recent research focuses on various aspects of this process, including methods of collecting, storing, and transmitting digital evidence.

Papers [1–5] use a legal approach to the collection and use of digital evidence in criminal proceedings in Ukraine, as well as to the consideration by courts of other categories of cases where digital evidence may also be used. The authors of these works focus on the distinction between the concepts of "electronic evidence" and "digital evidence", and also analyze court decisions on the recognition of such evidence as admissible and reliable. However, the issues of technical support for the integrity and authenticity of memory dumps during their transmission remain unaddressed. The author emphasizes the complexity of solving problems related to the admissibility of such evidence due to its technical nature, and suggests that the methods of collecting it should be referred to a separate procedural category, which, however, is not the subject of the present study.

The studies [6–10] refer to the growing role of digital evidence, mainly in criminal cases, and the problems with its recognition, which are common to most countries. However, while case law countries approach this issue with some flexibility, for the countries of the Romano-Germanic legal system, the issue of the relevance and admissibility of digital evidence is extremely acute.

There are also a number of studies [11–14] that analyze models of digital forensics processes in order to facilitate the choice of the most optimal one for investigative authorities, as well as propose an access control mechanism that includes the preservation of digital evidence, useful for the court in terms of ensuring the authenticity, confidentiality and reliability of digital evidence, and consider the issues of mobile forensics. In particular, study [13] presents a framework called the "Role-Based Mobile Forensic Framework with Cryptography (RBMF2C)" that can be used in forensic examination. The proposed platform consists of five layers: access control, evidence collection, data analysis layer, privacy and reporting.

However, these studies do not provide solutions for the mechanism of transporting digital evidence to a secure repository.

The study [15] considers the principles of organizing a secure data transfer interface based on the Tree-based ORAM M-ary algorithm. Although this study deals with general aspects of secure data transfer, its results can be applied to the development of secure methods for transferring memory dumps. Study [16] proposes a model for improving the entire process of investigating and obtaining digital evidence, analyzing the use of cryptography algorithms and hash functions. In papers [17–20], the focus shifts to the collection of digital evidence, a procedure that has acquired a systematic approach and structure, and deals with legal policy and technological aspects of procedural decision-making. For memory analysis, traditional forensic methods are being investigated, including signature-based methods, dynamic methods performed in a sandbox environment, and machine learning approaches [21].

However, specific developments aimed at ensuring the integrity and authenticity of digital evidence during its transfer require further development and justification. The authors do not provide practical recommendations for building the "secure corridor" for transporting memory dumps, and there is no comprehensive approach to the problem.

An analysis of scientific sources shows that despite significant progress in the study of the collection and use of digital evidence, the issue of secure transfer of memory dumps remains insufficiently studied. Most studies focus on the technical aspects of information extraction, but without further analysis of its integrity and reliability during transmission. The main reasons for this are the complexity of the technical implementation of secure transmission channels, the lack of standardized protocols, and the limited number of free tools to ensure data integrity and authenticity.

Thus, an unresolved problem is the development of effective and affordable technical solutions for the secure transportation of RAM dumps from the moment they are created to storage in specialized repositories. This includes the development of encryption, authentication and data integrity verification methods, as well as the creation of free tools that would ensure the secure transfer of digital evidence for further analysis in criminal and other legal proceedings. This limits the ability of law enforcement agencies and independent experts to examine digital evidence, including memory dumps.

The identified problems indicate the need to develop a secure architecture for the transfer of memory dumps, which will become the basis for improving the reliability and integrity of digital evidence in court proceedings, including in Ukraine.

The aim of research is to develop a technical approach to the secure transmission of memory dumps through secure channels for further analysis of digital evidence by digital forensics methods. This will make it possible to guarantee the authenticity, integrity and preservation of digital evidence, which can be used primarily in criminal, civil and administrative proceedings.

The research results will help to increase the level of security of digital evidence that can be used in court practice and create a free alternative solution for judicial and law enforcement agencies.

2. Materials and Methods

The object of research is the process of forming, transferring and storing RAM dumps within the framework of digital forensics.

The main hypothesis of the study is that the use of specialized cryptographic methods and authentication mechanisms can guarantee the integrity, reliability and security of the transferred memory dumps without the risk of their unauthorized modification or loss.

Assumptions made in this paper:

– obtaining memory dumps is carried out exclusively in compliance with digital forensics standards and in accordance with international practices (in particular, NIST and ISO 27037:2012);
– transfer of memory dumps should be carried out through secure communication channels using reliable encryption and authentication;
– open-source tools used in digital forensics, such as Volatility, LiME and Rekall, are used to analyze digital evidence.

Simplifications adopted in the paper:

– the study does not take into account the impact of the human factor (expert error) on the quality of digital evidence preservation;
– the analysis is carried out in a controlled environment without the influence of external attacks on the dump's transmission infrastructure.

The following scientific methods were used in the study:

1) *theoretical methods* – in analyzing scientific sources and regulations on digital forensics and information security; in comparing existing approaches to digital evidence preservation;

2) *empirical methods* – when modelling the processes of creating and transmitting memory dumps through secure communication channels; when testing cryptographic mechanisms and algorithms for monitoring the integrity of digital evidence;

3) *instrumental means* – when using digital forensics tools to obtain and analyze memory dumps (Volatility, LiME, Rekall); when using cryptographic libraries to encrypt and authenticate transmitted data (OpenSSL, GnuPG).

The study was conducted in a virtualized environment based on Linux and Windows operating systems, which allows testing methods for securely obtaining and transferring memory dumps in various

real-world application scenarios. To evaluate the effectiveness of the proposed solutions, test scenarios were used to simulate potential threats and verify the security of the transmitted digital evidence.

The study also analyzed existing state-of-the-art methods for obtaining and transmitting memory dumps used in digital forensics. The study included:

1) comparison of popular dumping tools (Volatility, Rekall, DumpIt);

2) analysis of standard file transfer methods (SCP, FTP, SMB) and their risks;

3) identification of key requirements for the secure transfer of digital evidence.

The results of the analysis showed that unsecured transmission methods can lead to data compromise, which reduces their forensic admissibility. Table 1 shows a comparison of existing methods for obtaining and transferring memory dumps.

Table 1

Comparison of methods for obtaining and transferring memory dumps (snapshots) (own research)

Method/tool	Operating systems	Type of dumping	Advantages	Disadvantages	Transmission methods	Transmission vulnerabilities
Volatility	Windows, Linux, macOS	Dump analysis (no creation)	Powerful analysis capabilities, support for multiple dump formats	Does not receive a dump directly, only analyses it	SCP, FTP, SMB	Possible to intercept data through unsecured channels
Rekall	Windows, Linux, macOS	Analysis and partial dumping	Deep memory analysis, support for OS profiles	High entry threshold, requires specialised knowledge	SCP, FTP, HTTP(S)	Possibility of MITM attacks when transmitting via HTTP(S)
DumpIt	Windows	Full memory dump creation	Ease of use, minimal system load	Works only on Windows, can leave digital traces	USB, SMB, FTP	No encryption when using FTP, risk of file substitution
FTK Imager	Windows	Full dump and preview capability	Graphical interface, meta information storage	High resource consumption	USB, FTP, SMB	No built-in encryption during transmission
LiME (Linux Memory Extractor)	Linux	Full memory dumping	Optimized for Linux, integration with third-party tools	Does not support Windows/macOS	SCP, Netcat	No encryption when using Netcat
AVML (Azure VM Memory Leak)	Linux	Creates memory dumps in virtual environments	Optimized for Azure and Linux systems	Does not work outside of Azure	SCP, HTTPS	High security, but dependent on cloud infrastructure

The key findings of the comparison are that SCP and HTTPS are the most secure methods of data transfer because they provide encryption. FTP and Netcat are unsafe for dump transfers because they lack built-in security. DumpIt and FTK Imager are the most convenient for Windows, but their transfer mechanisms require additional protection. LiME is the best choice for Linux, but it also requires additional security measures for data transfer.

3. Results and Discussion

3.1. Obtaining digital memory dumps (casts)

Obtaining memory dumps is a critical process in digital forensics, as they contain traces of malware, user activity history, and other potentially important evidence. As part of the study, let's implement a mechanism for securely obtaining RAM dumps in accordance with international standards. The main stages of implementation:

1) use of a proprietary Python script or specialized utilities Volatility and Rekall to obtain dumps, which ensures their compatibility with modern analysis methods;

2) generating a file's hash value immediately after creation to ensure its integrity;

3) automated logging of all actions to document the chain of custody of evidence, which is critical for its legitimacy in court proceedings.

The *Chain of Custody* documentation procedure, which ensures the integrity, reliability and authenticity of digital evidence from the moment it is collected to its use in court, includes:

1) documentation of each stage;

2) protection against unauthorized access;

3) physical and digital preservation;

4) tracking all changes;

5) readiness for forensic examination.

When analyzing memory dumps:

1. Obtaining a dump using forensic tools (Volatility, Rekall, Python script).

2. Calculating hash amounts to confirm authenticity.

3. Saving the original dump in a read-only environment.

4. Performing analysis on a copy only.

5. Keeping a log of all actions on digital evidence.

Thus, the general structure of a secure vault includes the following levels:

1) the level of collection and primary processing of digital evidence;

2) the level of encryption and storage;

3) the level of access control and user authentication. Each of these levels implements specific functions that provide comprehensive protection of digital evidence.

Cross-platform Python script for taking a memory dump after automatic OS detection:

```
import os
import platform
import subprocess

def dump_memory(output_file,
winpmem_path="winpmem.exe"):
  system = platform.system()

  if system == "Windows":
    print("[+]Windows is detected. Using winpmem...")
    subprocess.run([winpmem_path, output_file], check=True)

  elif system == "Linux":
    print("[+]Linux is detected. Using LiME...")
    lime_path = "/path/to/lime.ko"  # Point the right way
    subprocess.run(["sudo", "insmod", lime_path, f"path={output_
    file}"], check=True)

  elif system == "Darwin":  # macOS
    print("[+] MacOS is detected. Using osxpmem...")
    osxpmem_path = "/path/to/osxpmem"  # Point the right way
    subprocess.run(["sudo", osxpmem_path, "-o", output_file], check=True)

  else:
    print("[–] The operating system is not supported")
    return

  print(f"[+] Memory dump saved to {output_file}")
```

ISSN-L 2664-9969; E-ISSN 2706-5448

```
if __name__ == "__main__":
    output_file = input("Enter a file name to save the dump:")
    winpmem_path = input("Enter the path to winpmem.exe (or leave
    it blank for the default):") or "winpmem.exe"
    dump_memory(output_file, winpmem_path)
```

3.2. Identification of vulnerabilities and risks of compromising digital evidence

It has been established that the transmission of memory dumps through unprotected channels can lead to: data interception by intruders, modification of file contents during transmission, loss of meta-information important for forensic examination of electronic evidence.

It is proposed to use cryptographic protection and authentication to eliminate these risks. The scheme of threats during the transfer of digital evidence is shown in Fig. 1.

Transferring memory dumps to a secure analysis environment is an important step in digital forensics. As part of our research, let's implement a mechanism for the secure transport of digital evidence, which includes:

1) the use of the secure SCP protocol, which allows file transfers over an SSH connection with authentication;

2) additional encryption of files before transmission to ensure their confidentiality;

3) checking the hash value before and after transfer to ensure file integrity.

The implemented procedure prevents unauthorized interference with the transmission process and ensures the authenticity of digital evidence.

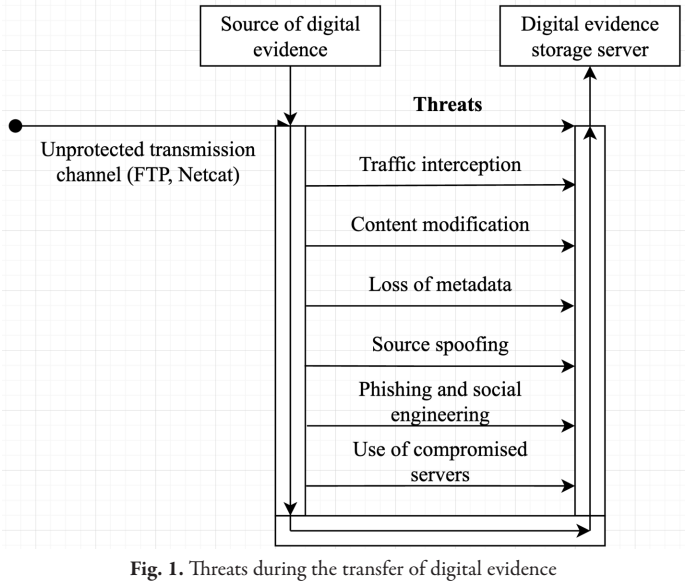

Fig. 1. Threats during the transfer of digital evidence

3.3. Investigation of cryptographic methods for protecting memory dumps

AES-256 is used to encrypt the stored memory dumps, which guarantees their protection in case of storage compromise. RSA is used to sign and verify dumps, ensuring the authenticity of digital evidence. The combination of these algorithms reduces encryption costs and allows signatures to be stored separately from the dumps themselves.

The following mechanisms are considered to ensure the protection of digital evidence:

1) use of symmetric and asymmetric encryption (AES-256, RSA);

2) protection during transmission using TLS and SSH protocols;

3) use of electronic signatures to guarantee the authenticity of files.

The study also confirmed the effectiveness of the combined use of these methods to protect the transferred memory dumps, as shown in Table 2.

The proposed architecture uses AES-256 for symmetric encryption of data before transmission, which provides a high level of cryptographic protection. Alternatively, for platforms that do not support AES-NI hardware acceleration, ChaCha20 can be used, which demonstrates high encryption speeds on ARM processors and mobile devices. AES-256 uses hardware acceleration (AES-NI) to compute AES-256, which significantly reduces CPU overhead.

However, for secure data transport, SSH alone is not a sufficient security measure, as it only provides session encryption, not data after transmission or storage. Therefore, the proposed architecture uses SSH in combination with SCP to securely copy files, and the files themselves are additionally encrypted with AES-256 in GCM mode before transfer, which guarantees their confidentiality even if the SSH connection is compromised. The practical implementation of this approach involves creating an encrypted memory dump before transmission and transporting it over SCP. On the recipient's side, the file's integrity is checked using SHA-256, which allows to detect any changes during the transfer process. To guarantee authenticity and avoid man-in-the-middle attacks, it is important to use a signature to verify the source. SSH is used exclusively as a transport mechanism for secure file transfer between the evidence collection server and the central repository. For an additional layer of security, tunneling over VPN with two-factor authentication is implemented on top of SSH.

The choice of these algorithms is based on their resistance to cryptographic analysis and efficiency in high-load systems. Computing efficiency is also important for systems that process large amounts of data (e. g., memory dumps). The computing resources required to implement encryption depend on the chosen algorithm. For AES-256 in GCM mode, when processing large amounts of data (e. g., memory dumps), the minimum requirements include a processor that supports AES-NI hardware acceleration (e. g., Intel Core i5 or AMD Ryzen 5) and 8 GB or more of RAM. For ChaCha20, which is effective on devices without AES hardware acceleration, a modern multi-core processor (ARM, x86_64) and at least 4 GB of RAM are sufficient to process memory dumps in real time.

Table 2

Characteristics of the studied cryptographic methods (own research)

Method	Type of encryption	Security level	Speed of execution	Resistance to attacks	Scope of application
AES-256	Symmetric	High	High	Resistant to known attacks, except for quantum attacks	Data protection on discs, VPN, Wi-Fi
RSA-4096	Asymmetric	High	Low	Vulnerable to quantum attacks	Signature protection, PKI, key transfer
TLS 1.3	Hybrid (AES, ChaCha20, RSA, ECDSA)	High	High	Protected against MitM, Forward Secrecy	Secure connection for data transmission
SSH (Ed25519)	Asymmetric	High	High	Resistant to key attacks	Remote access, file transfer
Electronic signature (ECDSA, EdDSA)	Asymmetric	High	Medium	Protected against forgery	Confirmation of the authenticity of documents and files

Thus, to securely transfer memory dumps, it is necessary to ensure that:

1) SSH is configured correctly (strong algorithms, no weak ciphers like AES-CBC);

2) the user uses SSH keys instead of a password;

3) transfer of the memory dump to a secure, isolated read-only environment;

4) additional file encryption is available before transfer (for example, AES-256 in GCM mode).

The code snippet below implements two main operations to ensure the security of digital evidence:

1) *encrypting the memory dump*. To protect data confidentiality, the AES-256 algorithm (via the Fernet library) is used to encrypt the memory dump before it is transmitted. This ensures that even if the data is intercepted during transmission, an attacker will not be able to access its contents;

2) *signature of the memory dump*. To ensure data integrity, the RSA signature mechanism is used. First, a hash of the memory dump is calculated using SHA-256, and then this hash is signed with the RSA private key. The signature allows the recipient to verify that the data has not been altered during transmission, ensuring the authenticity of the digital proof.

Thus, the combination of encryption (AES-256) and signature (RSA) guarantees both the confidentiality and integrity of the transmitted memory dumps.

A Python code snippet that encrypts a memory dump before transmission (AES-256-GCM):

```
from cryptography.hazmat.primitives.asymmetric import rsa, padding
from cryptography.hazmat.primitives import hashes, serialization
from cryptography.fernet import Fernet

# RSA key generation (for testing)
private_key = rsa.generate_private_key(
  public_exponent=65537,
  key_size=2048
)
public_key = private_key.public_key()

# Key serialisation (if required)
private_pem = private_key.private_bytes(
  encoding=serialization.Encoding.PEM,
  format=serialization.PrivateFormat.TraditionalOpenSSL,
  encryption_algorithm=serialization.NoEncryption()
)

public_pem = public_key.public_bytes(
  encoding=serialization.Encoding.PEM,
  format=serialization.PublicFormat.SubjectPublicKeyInfo
)

# Memory dump signature function
def sign_dump(dump_data, private_key):
  hash_value = hashes.Hash(hashes.SHA256())
  hash_value.update(dump_data)
  digest = hash_value.finalize()

  signature = private_key.sign(
    digest,
    padding.PSS(
      mgf=padding.MGF1(hashes.SHA256()),
      salt_length=padding.PSS.MAX_LENGTH
    ),
    hashes.SHA256()
  )
  return signature

# Signature verification function
def verify_signature(dump_data, signature, public_key):
  hash_value = hashes.Hash(hashes.SHA256())
  hash_value.update(dump_data)
  digest = hash_value.finalize()

  try:
    public_key.verify(
      signature,
      digest,
      padding.PSS(
        mgf=padding.MGF1(hashes.SHA256()),
        salt_length=padding.PSS.MAX_LENGTH
      ),
      hashes.SHA256()
    )
    return True
  except:
    return False

# AES-256 key generation for encryption
key = Fernet.generate_key()
cipher = Fernet(key)

# Reading a memory dump
with open(dump_file, "rb") as f:
  dump_data = f.read()

# Signing the dump
signature = sign_dump(dump_data, private_key)

# Dump encryption
encrypted_data = cipher.encrypt(dump_data)

# Saving an encrypted dump and signature
with open(dump_file + ".enc", "wb") as f:
  f.write(encrypted_data)

with open(dump_file + ".sig", "wb") as f:
  f.write(signature)
```

A hash value verification procedure is used to confirm the integrity of digital evidence after it has been received and transmitted. Verification is based on the following steps:

1) generation of a hash value (SHA-256) before transmission, which allows to record the initial state of the file;

2) obtaining the hash value on the server side after the transfer, which allows confirming its immutability;

3) comparing the obtained values to identify possible changes and ensure the reliability of the digital proof.

Hash value matching is a key criterion for maintaining the authenticity of evidence, and deviations may indicate attempts to interfere with the transfer process or damage files (Fig. 2, 3).

As can be seen from the figures (Fig. 2, 3), the hash sums before and after the memory dump transfer are completely the same and are: cade2fd543eac1cd2eab9ccd0a840d83481a3f00e16015287323b2cb44fe0686.

For the secure transfer of memory dumps, it is possible to use the SCP (Secure Copy Protocol) protocol, which ensures secure file transfer over an SSH connection. SCP allows data to be transferred with encryption, which guarantees the confidentiality of the transferred files and protects them from interception and modification during transport.

ISSN-L 2664-9969; E-ISSN 2706-5448

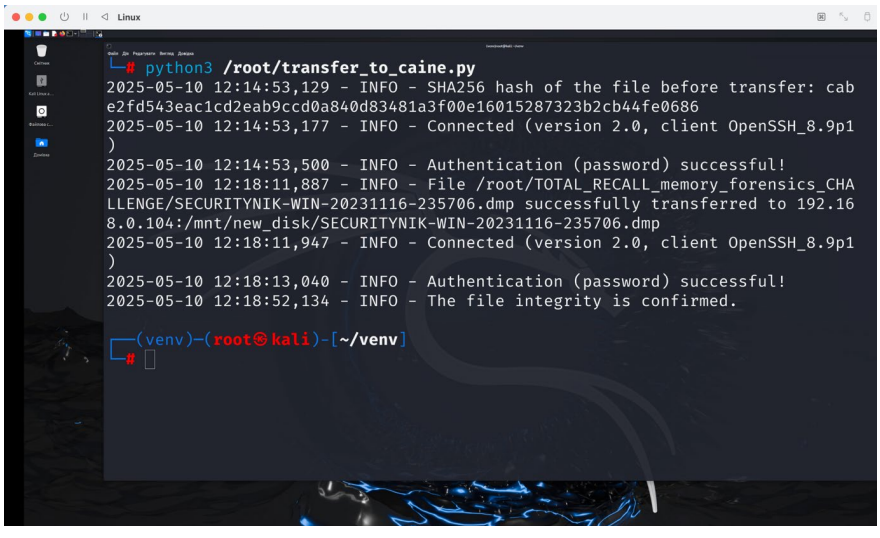

Fig. 2. Screenshot of code execution on Kali Linux to transfer a file via SSH: file hash count before transfer, successful password entry, file integrity check after transfer

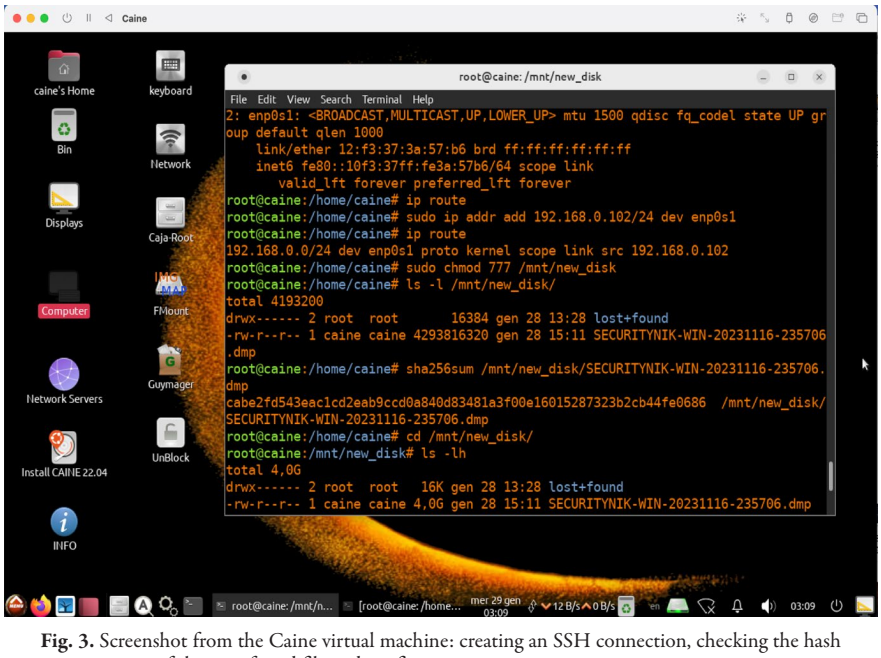

Fig. 3. Screenshot from the Caine virtual machine: creating an SSH connection, checking the hash of the transferred file and confirming its presence in a secure container

As part of the "secure corridor" mechanism, SCP was used in combination with additional security measures: pre- and post-transfer hash values (SHA-256) to confirm the integrity of the file and restricted access to the storage environment ("read-only" container in Caine). This approach minimizes the risk of compromising the transferred evidence and ensures that the files received on the receiver's side remain identical to the original.

It is important to note that the SCP protocol itself was not modified during the study, but its use within the proposed architecture was supplemented by integrity control and tampering protection mechanisms.

Research results: during the experimental study, the process of securely transferring a memory dump between two environments was tested: a data source (Kali Linux) and a digital evidence analysis environment (Caine). The main results primarily include verification of the security of the connection. It was confirmed that the use of the SSH protocol provides a secure file transfer channel without the risk of interception.

Secondly, the integrity of the transferred data was confirmed. The hash sums of the memory dump before and after the transfer (Fig. 2, 3) were completely identical, which confirms that the file was not modified during the transfer. Random checks of different memory dumps also showed a 100% hash match, which confirms the reliability of the verification mechanism.

Thirdly, the proper functioning of the created secure storage was proved. The file transferred to the Caine environment was successfully placed in a read-only container, which makes it impossible to modify it. Commands to check the status of the file system (ls, mnt) showed that the saved file is read-only, with no possibility of modification or deletion.

Fourth, a comparison with other methods proves that:

1) unlike standard transmission through unsecured channels, the use of the "secure corridor" guarantees that data cannot be altered during transmission, which is critical for digital forensics;

2) the use of logging of all operations in the Caine environment provides the ability to further analyze and record any actions with copies of the transferred files.

3.4. Model of a "safe corridor" for transporting digital evidence to a storage facility ("read-only") for storage and further analysis of transferred memory dumps

Based on the results obtained, the "secure corridor" mechanism was implemented, which includes:

1) a secure connection between the source and the storage server;

2) automated verification of hash values before and after transfer;

3) logging of all actions in the evidence storage system.

This is important because memory dumps are usually taken during the investigation phase, as this data may contain important evidence, such as processes, network connections, or malware remnants that can be used in a lawsuit.

To implement the "secure corridor" mechanism, several important stages of data transmission and integrity verification have been developed. Fig. 2 shows the result of running the code that implements the process of transferring a file over a secure connection. Important elements of this process are:

- calculating the file hash before transferring;
- using an SSH connection for secure transfer;
- successfully entering a password and confirming the file transfer;
- confirming the file's integrity after transfer. This stage is important for ensuring the security of data transfer, as it allows to ensure that the file will not be altered during transmission.

Fig. 3 demonstrates the process of retrieving a file in the Caine environment through the established SSH connection. After the file is successfully transferred, the Caine virtual machine uses a read-only container to store the file, which significantly reduces the possibility of unauthorized access. The machine also checks the hash of the transferred file to ensure its integrity. In addition, the mnt utility shows the presence of the transferred file in the storage, which confirms the success of the process.

To ensure the confidentiality and integrity of the transmitted data, the secure corridor mechanism uses cryptographic algorithms at several levels. Before a memory dump is transmitted, its hash value is calculated using SHA-256, which allows to check its integrity after transportation. The communication channel is protected by the SSH protocol, which uses symmetric encryption (AES-256 or ChaCha20) to ensure data confidentiality. Additionally, SSH authentication can be performed using RSA or ECDSA, which prevents unauthorized access to transmission and storage servers. This multi-level scheme ensures that the transmitted digital evidence is not altered or compromised during transport.

These steps are critical to the functioning of the secure corridor mechanism, as they confirm the integrity of the data during transmission and ensure that it is not altered during transmission between systems. They also demonstrate the implementation of the secure corridor for the transfer of digital evidence, which includes not only secure SSH transfer, but also additional steps to verify the integrity of files and create a secure environment for storing the received data. To ensure the security of the file transfer process, the SCP protocol is used over an SSH connection, which guarantees data encryption during transmission and protects against interception or modification of files in transit.

Changes in the data transfer process, as shown in Fig. 2, 3, confirm the correct implementation of the "secure corridor" mechanism, which includes checking the integrity of files before and after transfer over a secure connection, as well as the availability of a secure environment for storing data on the recipient's side.

Custom code (Python) that implements the secure corridor:

```
import os
import hashlib
import paramiko
from scp import SCPClient
import logging

# Logging settings
logging.basicConfig(level=logging.INFO,
format="%(asctime)s – %(levelname)s – %(message)s")

# Function for calculating SHA256 hash
def calculate_hash(file_path):
    sha256_hash = hashlib.sha256()
    try:
        with open(file_path, "rb") as f:
            for byte_block in iter(lambda: f.read(4096), b""):
                sha256_hash.update(byte_block)
        return sha256_hash.hexdigest()
    except Exception as e:
        logging.error(f"Error computing hash for {file_path}: {e}")
        return None

# File transfer function via SCP
def transfer_file_to_caine(local_path, remote_path, hostname,
username, password):
    try:
        # Setting up an SSH connection
        ssh = paramiko.SSHClient()
        ssh.set_missing_host_key_policy(paramiko.AutoAddPolicy())
        ssh.connect(hostname, username=username,
password=password)

        # Using SCP to transfer a file
        with SCPClient(ssh.get_transport()) as scp:
            scp.put(local_path, remote_path)
        logging.info(f"File {local_path} successfully transferred to
        {hostname}:{remote_path}")
        ssh.close()
    except Exception as e:
        logging.error(f"Error transferring file: {e}")

# Main function
def main():
    # The path to the memory dump
    dump_file = "TOTAL_RECALL_memory_forensics_CHALLENGE/
    SECURITYNIK-WIN-20231116-235706.dmp" # Path to the
    memory dump in the target machine
    caine_path = "/mnt/new_disk/" # Directory on CAINE where to
    transfer the memory file

    # Data to connect to CAINE
    hostname = "192.168.0.102" # CAINE IP address
    username = "caine"       # Username in CAINE
    password = "12345678" # Password to connect to CAINE

    # Checking the existence of a file
    if not os.path.exists(dump_file):
        logging.error(f"File {dump_file} not found!")
        return

    # Calculating hash
    hash_before = calculate_hash(dump_file)
    if not hash_before:
        logging.error("The hash could not be calculated. End of work.")
        return

    logging.info(f"SHA256
: {hash_before}")

    # File transfer
    transfer_file_to_caine(dump_file, os.path.join(caine_path, os.path.
    basename(dump_file)), hostname, username, password)

    # Additional hash verification after transfer
    try:
        ssh = paramiko.SSHClient()
        ssh.set_missing_host_key_policy(paramiko.AutoAddPolicy())
        ssh.connect(hostname, username=username,
        password=password)

        remote_file_path = os.path.join(caine_path, os.path.
        basename(dump_file))
        stdin, stdout, stderr = ssh.exec_command(f"sha256sum {remote_
        file_path}")
        hash_after = stdout.read().decode().split()[0]
        ssh.close()

        if hash_before == hash_after:
            logging.info("The file integrity is confirmed.")
        else:
            logging.warning("Hash does not match! The file is corrupted.")
    except Exception as e:
        logging.error(f"Error checking hash against CAINE: {e}")

if __name__ == "__main__":
    main()
```

Thus, within the framework of the study, the architecture of a secure digital evidence repository was developed, which includes:

1) the use of a read-only file system container, which ensures that the original files cannot be altered;

ISSN-L 2664-9969; E-ISSN 2706-5448

2) physical separation of the storage from the working environment to prevent compromise of digital evidence;

3) access control using two-factor (multi-level) authentication;

4) logging of all actions with digital evidence, which allows tracking any access attempts and integration of a mechanism for logging all operations with files.

The use of a read-only container protects the data from modification, which can be important for the legal admissibility of digital evidence. This reduces the risk of data compromise and ensures high reliability of evidence in court proceedings. Compared to traditional methods of storing digital evidence, the proposed approach using the secure corridor and read-only containers offers an additional level of protection against possible changes and access, which significantly increases the reliability of data storage.

Thus, the proposed model of transmission and storage allows to ensure the maximum level of security and protection against modification of digital evidence and its further suitability for forensic examinations. It complies with international and national standards of digital forensics and can be used in real-world conditions.

3.5. Evaluation of the effectiveness of the proposed approach based on process modelling in the test environment

To evaluate the effectiveness of the proposed approach, let's conduct testing in a specially created test environment:

1. A read-only storage was created physically separated from the working environment, which guarantees the impossibility of modifying the original files (Fig. 4, 5).

2. The collected memory dumps were transferred using our own Python script using the "secure corridor" from the Kali Linux virtual machine to the Caine virtual machine to the created container in read-only mode. The integrity of the files after transportation and storage was checked by comparing the hash sum.

3. After obtaining and saving the memory dumps, a detailed analysis was carried out to identify digital artefacts that may be of evidentiary value.

The following tools and methods were used for this purpose:

– Volatility – to analyze active processes, network activity and connected devices;

– Rekall – to search for malicious code, hidden processes and changes in system tables;

– specialized Python scripts – for automated metadata analysis and identification of suspicious objects, etc.

Further analysis is also possible with Autopsy, which allows to connect memory dumps as data sources (although it does not provide full analysis like Volatility or Rekall). If the memory snapshot contains extracted files (e. g., .exe, .dll, .txt), Autopsy can analyze them for malicious code, hashes, and metadata. Through Autopsy modules, it is possible to search for browser activity artefacts, chat traces, credentials, extract text information, perform keyword searches or check process traces as found artefacts, and automatically compare the found files in the memory dump with NIST or other hash databases to identify known threats or critical files.

Rekall (Rekall Memory Forensics Framework) is used to analyze memory dumps. The utility allows to examine saved processes, drivers, network connections, file residues and other artefacts in RAM, supports analysis of Windows, Linux and macOS memory dumps, various dump formats, and is best at analyzing user sessions, recovering deleted data, and detecting rootkits.

The test results confirmed the reliability of the implemented system and the absence of changes in the transferred files. The analysis showed that the proposed methodology allows obtaining important digital evidence that can be used in court investigations. The test results are presented in Table 3.

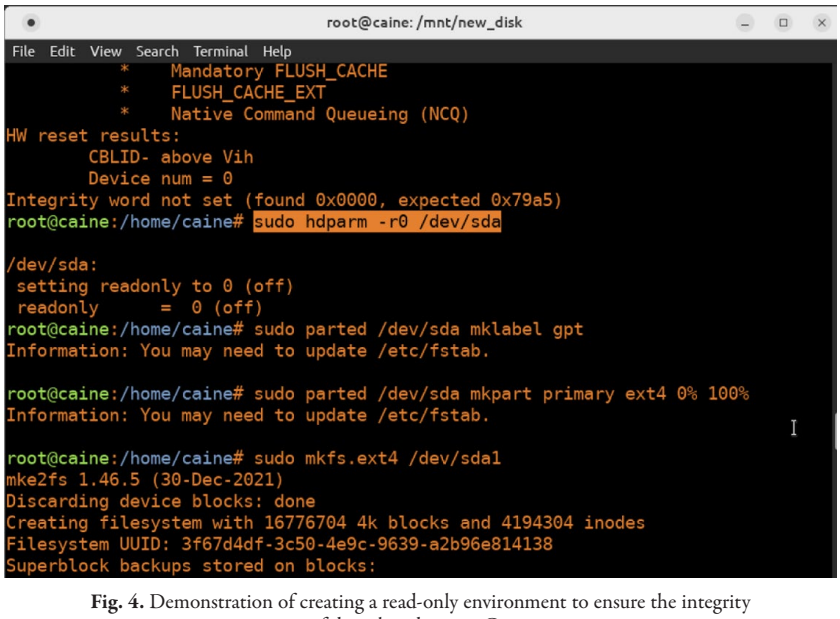

Fig. 4. Demonstration of creating a read-only environment to ensure the integrity of digital evidence at Caine

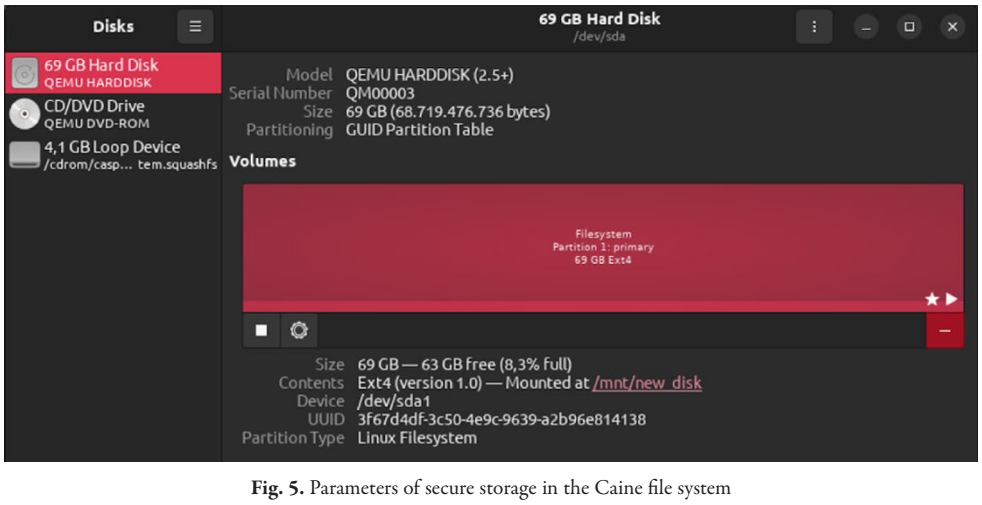

Fig. 5. Parameters of secure storage in the Caine file system

Table 3

Test results, including integrity checks, transfer speeds, and efficiency of memory dump analysis

Test parameter	Test method	Result
Transferring a dump via SCP	Transmission time (sec)	12.4
File integrity check (SHA-256)	Hash value matching	Yes
Analysis of active processes (Volatility)	Abnormal processes are detected	3
Analysis of network activity (Volatility)	Suspicious connections are detected	2
Detection of hidden processes (Rekall)	The number of hidden processes found	1
Analysis of extracted files (Autopsy)	Malicious files are detected	2
Time for complete analysis of the dump (min)	Average time	18.7

Thus, it has been confirmed that the proposed architecture provides a high level of security during the acquisition, transmission, storage and analysis of memory dumps. The developed approach allows preserving the integrity and authenticity of digital evidence, which is critical for digital forensics.

3.6. Discussion

An analysis of existing methods of obtaining and transmitting memory dumps has shown that most modern solutions do not provide a sufficient level of security when transmitted over unsecured channels. This is due to the lack of built-in encryption mechanisms in some tools, as well as significant risks of data loss during transportation.

The identified vulnerabilities justify the need to use additional security mechanisms, in particular, cryptographic methods. The development and detailed analysis of the proposed "secure corridor" confirmed that the use of an SSH connection significantly reduces the risks of unauthorized access and modification of data during the transportation phase.

The developed model for the transfer of digital evidence and the storage architecture in general allows to ensure its integrity through the use of SCP, a read-only storage file system and a mechanism for logging operations.

The proposed model for secure storage and transfer of digital evidence is called the *Secure Memory Dump Framework (SMDF)*. It includes three main components:

1) *secure evidence collection environment* – implemented through a specially isolated environment, which guarantees that the received memory dumps are not modified;

2) *secure transfer mechanism* – uses SCP with hash control (SHA-256) to verify the integrity of the transferred files;

3) *secure storage* – a read-only file environment that makes it impossible to modify or tamper with digital evidence.

Unlike traditional methods of transferring digital evidence, the proposed SMDF approach has several advantages:

1) unlike FTP or SMB, the use of SCP provides cryptographic protection of the transfer;

2) unlike standard file servers, the read-only file environment completely eliminates the risk of modification of evidence;

3) the built-in hash control mechanism allows to confirm the authenticity of data without the need for additional checks.

The "secure corridor" mechanism in the SMDF system guarantees a high level of security through the use of the SCP protocol for data transmission. This approach provides the following security services:

1) confidentiality, because through the use of the SSH cryptographic protocol, SCP protects data from unauthorized access during its transmission over the network. This significantly reduces the risk of sensitive information leakage compared to unencrypted protocols such as FTP;

2) integrity, as the built-in hash control mechanism at each stage of data transmission allows to verify its authenticity and integrity. Each transferred memory dump is accompanied by a unique hash, which confirms that the data has not been modified during the transfer;

3) accessibility (provable authenticity), as the implementation of read-only access and multi-factor authentication minimizes the risks of unauthorized access and data loss. This ensures stable access to digital evidence at any time without the threat of modification, in accordance with the *Chain of Custody* procedure.

These security aspects determine the high efficiency of the proposed SMDF approach for the transfer of digital evidence within the forensic analysis of memory dumps. A detailed visual comparison of the SMDF solution with other existing methods is shown in Table 4.

Table 4

Comparison of approaches to ensuring the security and integrity of digital evidence in the forensic analysis of memory dumps

Approach/Solution	Integrity of evidence	The level of security	Performance	Compliance with standards	Cost	Disadvantages
FTP (File Transfer Protocol)	Low; no built-in integrity checking mechanisms	Low; transmission in the open	High; fast transmission	Does not meet modern standards	Free	High risk of data modification and interception
SFTP (Secure File Transfer Protocol)	High; uses SSH to ensure integrity	High; data encryption	Medium; impact of encryption	Compliant with standards such as ISO/IEC 27001	Free (depends on infrastructure)	Requires key management and authentication
Copy to physical media (USB, HDD, SSD)	High; minimal risk of damage	Low; risk of loss or theft	High; fast copying	Lack of clear standards	The cost depends on the media	Vulnerability to physical loss, third-party access
Cloud storage with encryption (e.g. zero disclosure)	High; cryptographic protection	High; data is encrypted before uploading	Average; depends on internet speed	Compliant with standards (GDPR)	Subscription or depends on the provider	Dependence on service, risks of trust in the supplier
Secure Memory Dump Framework (SMDF)	Very high; checksums, digital signatures, change log	Very high; multi-level encryption, access auditing, access controls	High; optimized for analyzing memory dumps	Compliant with NIST 800-86, ISO 27037	Free	Requires customization and integration with other systems

ISSN-L 2664-9969; E-ISSN 2706-5448

Thus, as can be seen from the comparative table, SMDF has advantages, the proposed solution is unique and useful, but most importantly, it allows for a full reproduction of the *Chain of Custody* procedure, which is critical for using the results of memory dump analysis in court proceedings.

Performance evaluation in the test environment also confirmed that the proposed solution meets international standards for digital forensics and can be used in real-world applications.

Unlike existing commercial methods, which often do not provide encryption at the transmission stage (e. g., FTP or SMB), the proposed "secure corridor" integrates cryptographic mechanisms that ensure data integrity and confidentiality. In addition, adding multi-factor authentication to the storage access process significantly reduces the risk of unauthorized interference.

Compared to similar approaches, the proposed read-only storage model eliminates the possibility of accidental or intentional modification of digital evidence, which is an important difference from conventional file servers or cloud storage without such restrictions.

The results obtained are directly aimed at solving the problems identified in the literature review. In particular, the issue of secure data transmission is solved by cryptographic protection and the selected transmission channel, and the problem of evidence integrity is solved by implementing a read-only environment in accordance with the *Chain of Custody* procedure. Thus, the results of the study make it possible to eliminate the main threats of compromising digital evidence.

Although the proposed approach guarantees a high level of security, it has certain limitations. In particular, the use of cryptographic encryption can affect system performance when processing large amounts of data. Implementation of multi-factor authentication may complicate access for emergency analysis. Implementing a read-only architecture may require additional resources and technical support. One of the limitations of the study is that the system was tested under controlled conditions, which may differ from real-world use cases. In addition, the study focuses mainly on software security aspects and does not cover potential hardware threats.

In criminal proceedings, the analysis of memory dumps plays an important role in the investigation of many types of crimes, especially under martial law. This method allows to identify and document evidence of treason, collaboration, espionage and cyber-sabotage activities. Analysis of RAM allows to recover deleted or encrypted messages, network connection history, and other digital artefacts that may indicate the transfer of data to hostile entities. In addition, it can detect the use of hidden malware that can be used for cyber espionage or remote control of devices to attack critical infrastructure. It is through the rapid collection and processing of digital evidence that threats to national security can be responded to promptly and perpetrators can be brought to justice.

A possible area for further development is the integration of machine learning mechanisms to automatically detect anomalies during the transmission and storage of digital evidence. In addition, the research can be extended to adapt the proposed architecture to distributed computing environments and cloud platforms, which will improve the scalability and availability of the system.

Thus, the research not only addresses the key issues of digital evidence security, but also opens up prospects for further improvement of information security systems within digital forensics and amendments to Ukrainian legislation on the use of digital evidence in court proceedings.

4. Conclusions

The analysis of modern methods of collecting, storing and processing digital evidence, namely memory dumps, has shown that existing approaches do not provide the appropriate level of efficiency and security required by the current situation in Ukraine. The proposed architecture of a secure digital evidence repository takes into account the current threats and needs of judicial, in particular criminal, proceedings in Ukraine and lays the technical basis for further improvement of legislation:

1. The study analyzed modern methods of obtaining and transferring memory dumps used in digital forensics. The results of the analysis showed that unprotected transmission methods can lead to data compromise, which reduces their judicial admissibility, so it is critical to comply with the Chain of Custody procedure. The authors also proposed their own cross-platform Python script for taking a memory dump after automatic OS detection.
2. Vulnerabilities and risks of compromising digital evidence are identified. The use of cryptographic protection and authentication to eliminate them is proposed. It is proved that the implemented mechanism of secure transportation of digital evidence makes it impossible to unauthorizedly interfere with the transmission process and ensures the preservation of their authenticity.
3. The known cryptographic methods of protection are investigated; the effectiveness of their combined use is confirmed. It is established that secure transfer of memory dumps requires correct SSH configuration, the use of SSH keys instead of a password, transfer of the memory dump to a secure, isolated read-only environment, and additional encryption of the file before transfer (the author's own code snippet is presented). The paper demonstrates the compliance of the obtained hash values with the model, which is a key criterion for preserving the authenticity of evidence.
4. The author provides the Python code of her own model of the "secure corridor" for transporting digital evidence to the "read-only" storage for storing and further analyzing the transferred memory dumps. In general, the architecture of a secure digital evidence repository has been developed, which includes: the use of a read-only file system container; access control at each level; logging of all actions with digital evidence, which guarantees the impossibility of modifying the original files. The proposed model complies with international and national standards of digital forensics and can be used in real life.
5. The effectiveness of the proposed approach is evaluated based on process modelling in a test environment. In particular, the collected memory dumps were transferred using our own Python script using the "secure corridor" from the Kali Linux virtual machine to the Caine virtual machine to the created container in read-only mode. The integrity of the files after transportation and storage was checked by comparing the hash sum. After obtaining and saving the memory dumps, they were analyzed in detail to identify digital artefacts that could be of evidentiary value.

The test results confirmed the reliability of the implemented system and the absence of changes in the transferred files. Thus, the proposed methodology allows obtaining important digital evidence that can be used in court investigations.

Conflict of interest

The author declares that she has no conflict of interest in relation to this research, including financial, personal, authorship or other, which could affect the research and its results presented in this article.

Financing

The research was conducted without financial support.

Data availability

All data are available in digital or graphical form in the main text of the manuscript.

Use of artificial intelligence

The author confirms that she did not use artificial intelligence technologies in the creation of the current work.

References

1. Avdeeva, G., Żywucka-Kozlowska, E. (2023). Problems of Using Digital Evidence in Criminal Justice of Ukraine and the USA. *Theory and Practice of Forensic Science and Criminalistics, 30 (1),* 126–143. https://doi.org/10.32353/khrife.1.2023.07
2. Romaniuk, V. V., Ablamskyi, S. Ye. (2024). Criteria for the admissibility of digital (electronic) evidence in criminal proceedings. *Law and Safety, 93 (2),* 140–150. https://doi.org/10.32631/pb.2024.2.13
3. Garasymiv, O., Marko, S., Ryashko, O. (2023). Digital evidence: some problematic issues regarding its concept and use in criminal justice. *Uzhhorod National University Herald. Series: Law, 2 (75),* 158–162. https://doi.org/10.24144/2307-3322.2022.75.2.25
4. Sezonov, V. S., Piddybna, A. V. (2021). Problematic issues of collection and use of documents as sources of evidence in criminal proceedings. *Scientific Notes of Taurida National V. I. Vernadsky University. Series: Juridical Sciences, 1,* 111–119. https://doi.org/10.32838/tnu-2707-0581/2021.1/20
5. Slipeniuk, T., Yankovyi, M., Nikitenko, V., Manzhai, O., Tiuria, Y. (2024). Problematic Issues of Using Electronic Evidence in Criminal Proceedings (SDG's). *Journal of Lifestyle and SDGs Review, 4 (1),* e01867. https://doi.org/10.47172/2965-730x.sdgsreview.v4.n00.pe01867
6. Dodge, A. (2017). The digital witness: The role of digital evidence in criminal justice responses to sexual violence. *Feminist Theory, 19 (3),* 303–321. https://doi.org/10.1177/1464700117743049
7. Watney, M. (2009). Admissibility of Electronic Evidence in Criminal Proceedings: An Outline of the South African Legal Position. *Journal of Information, Law & Technology, 1.* Available at: http://go.warwick.ac.uk/jilt/2009_1/watney
8. Bharati, R., Khodke, P. G., Khadilkar, C. P., Bawiskar, Dr. S. (2024). Forensic Bytes: Admissibility and Challenges of Digital Evidence in Legal Proceedings. *International Journal of Scientific Research in Science and Technology, 11 (16),* 24–35. https://doi.org/10.2139/ssrn.4896874
9. Casey, E. (2011). *Digital Evidence and Computer Crime Forensic Science: Computers and the Internet.* Academic Press, 837. Available at: https://rishikeshpansare.wordpress.com/wp-content/uploads/2016/02/digital-evidence-and-computer-crime-third-edition.pdf
10. Hewling, M., Sant, P. (2012). *Digital Forensics: An integrated approach. Conference: CFET At: Canterbury.* Available at: https://www.researchgate.net/publication/259055528_Digital_Forensics_An_integrated_approach
11. Abulaish, M., Haldar, N. A. H. (2018). Advances in Digital Forensics Frameworks and Tools. *International Journal of Digital Crime and Forensics, 10 (2),* 95–119. https://doi.org/10.4018/ijdcf.2018040106
12. Abulaish, M., Haldar, N. A. H., Jahiruddin, J. (2021). P2DF. *International Journal of Digital Crime and Forensics, 13 (6),* 1–15. https://doi.org/10.4018/ijdcf.288547
13. Hyder, M. F., Arshad, S., Arfeen, A., Fatima, T. (2022). Privacy preserving mobile forensic framework using role-based access control and cryptography. *Concurrency and Computation: Practice and Experience, 34 (23).* https://doi.org/10.1002/cpe.7178
14. Vaddi, K. S., Kamble, D., Vaingankar, R., Khatri, T., Bhalerao, P. (2024). Enhancements in the world of digital forensics. *IAES International Journal of Artificial Intelligence (IJ-AI), 13 (1),* 680. https://doi.org/10.11591/ijai.v13.i1.pp680-686
15. Kozak, Ye. B. (2021). Organization of a secure data transmission interface based on the M-ary algorithm "tree-based oram". *Scientific Notes of Taurida National V. I. Vernadsky University. Series: Technical Sciences, 4,* 84–89. https://doi.org/10.32838/2663-5941/2021.4/13
16. Abbas, T., Altaher, A. (2021). Identifying Digital Forensic Frameworks Based on Processes Models. *Iraqi Journal of Science, Special Issue,* 249–258. https://doi.org/10.24996/ijs.2021.SI.1.35
17. Kasper, A. (2015). *Multi-level analytical frameworks for supporting cyber security legal decision making.* [Doctoral Thesis; Estonian Business School]. Available at: http://ebs.ee/en/research-and-doctoral-studies/publications/phd-theses
18. Joseph, D. P., Viswanathan, P. (2022). A *Comprehensive Survey and Analysis on Multi-Domain Digital Forensic Tools, Techniques and Issues.* https://doi.org/10.21203/rs.3.rs-1988841/v1
19. Saharan, S., Yadav, B. (2022). Digital and Cyber Forensics: A Contemporary Evolution in Forensic Sciences. Crime Scene Management within Forensic Science. *Crime Scene Management within Forensic Science, Forensic Techniques for Criminal Investigations,* 267–294. https://doi.org/10.1007/978-981-16-6683-4_11
20. Kapade, P., Pandey, A. K. (2018). Technical issues & challenges in memory forensics. *International Journal of Creative Research Thoughts (IJCRT), 6 (2),* 1617–1621. Available at: https://ijcrt.org/papers/IJCRT1812859.pdf
21. Nyholm, H., Monteith, K., Lyles, S., Gallegos, M., DeSantis, M., Donaldson, J., Taylor, C. (2022). The Evolution of Volatile Memory Forensics. *Journal of Cybersecurity and Privacy, 2 (3),* 556–572. https://doi.org/10.3390/jcp2030028

Maryna Larchenko, *PhD, Department of Cybersecurity and Mathematical Simulation, Chernihiv Polytechnic National University, Chernihiv, Ukraine; Department of Political Science, Law and Philosophy, Nizhyn Mykola Gogol State University, Nizhyn, Ukraine, e-mail: urlinka2006@gmail.com, ORCID: https://orcid.org/0000-0002-2643-980X*