Forensic Science International: Digital Investigation 44 (2023) 301486

Contents lists available at ScienceDirect

Forensic Science International: Digital Investigation

journal homepage: www.elsevier.com/locate/fsidi

A framework for live host-based Bitcoin wallet forensics and triage

Arran Holmes, William J. Buchanan*

Blockpass ID Lab, Edinburgh Napier University, Edinburgh, UK

ARTICLE INFO

Article history:
Received 30 March 2022
Received in revised form
7 November 2022
Accepted 29 November 2022
Available online xxx

Keywords:
Cryptocurrency
Bitcoin
Wallet
Triage
Asset recovery
Forensics

ABSTRACT

Organised crime and cybercriminals use Bitcoin, a popular cryptocurrency, to launder money and move it across borders with impunity. The UK and other countries have legislation to recover the proceeds of crime from criminals. Recent UK case law has recognised cryptocurrency assets as property that can be seized and realised under the Proceeds of Crime Act (POCA). To seize a cryptocurrency asset generally requires access to the private key. Anecdotal evidence suggests that if cryptocurrency is not seized quickly after enforcement action has taken place, it will be transferred to other wallets making it difficult to seize at a future time. We investigate how Bitcoin could be seized from an Electrum or Ledger hardware wallet, during a law enforcement search, using live forensic techniques and a dictionary attack.

We conduct a literature review examining the state-of-the-art in Bitcoin application forensics and Bitcoin wallet attacks. Concluding, that there is a gap in research on Bitcoin wallet security and that a significant proportion of the available literature comes from a small group of academics working with industry and law enforcement (Volety et al. 2019; Van Der Horst et al., 2017; Zollner et al., 2019). We then forensically examine the Electrum software wallet and the Ledger Nano S hardware wallet, to establish what artefacts can be recovered to assist in the recovery of Bitcoin from the wallets. Our main contribution is a proposed framework for Bitcoin forensic triage, a collection tool to recover Bitcoin artefacts and identifiers, and two proof of concept dictionary-attack tools written in Python and OpenCL.

We then evaluate these tools to establish if an attack is practicable using a low-cost cluster of public cloud-based Graphics Processing Unit (GPU) instances. During our investigation, we find a weakness in Electrum's storage of encrypted private keys in RAM. We leverage this to make around 2.4 trillion password guesses. We also demonstrate that we can conduct 16.6 billion guesses against a password protected Ledger seed phrase.

1. Introduction

Criminals are turning to Bitcoin to launder money and move it across borders. This is because the UK and many other countries have strict regulations on fiat currencies. In the UK, financial institutions have to make a Suspicious Activity Report (SAR) if they suspect criminal activity. Almost half a million SARs are submitted every year (National Crime Agency 2021). Where a financial institution is dealing with funds that they suspect are criminal property, they will request a Defence Against Money Laundering (DAML) from the National Crime Agency (NCA). This notification allows UK law enforcement to investigate the criminal property and take enforcement action relating to the criminal property, possibly resulting in its seizure (National Crime Agency 2019). However, cryptocurrencies like Bitcoin are largely unregulated. This means that criminals can avoid being identified committing money laundering offences which include concealing, disguising, converting, transferring or removing criminal property from England, Wales, Scotland or Northern Ireland.

This paper aims to explore forensic opportunities that law enforcement can leverage while executing a search warrant. The goal is to identify the use of Bitcoin and recover the private key, thereby gaining control of the asset for seizure under the Proceeds of Crime Act (POCA). Its main contribution is defining a methodology that could be used in real-life Bitcoin investigations, and aims to provide guidance for the costs involved.

1.1. Background

One of the key legislative tools UK law enforcement have to deal with criminal property is the Proceeds of Crime Act, 2002 (POCA).

* Corresponding author.
E-mail address: w.buchanan@napier.ac.uk (W.J. Buchanan).

https://doi.org/10.1016/j.fsidi.2022.301486

This legislation gives UK law enforcement the power to seize criminal property, which includes cryptocurrency assets. UK law enforcement is well equipped to seize fiat currency and traditional assets under this legislation but cryptocurrency assets can be difficult to gain control of.

UK law enforcement targets criminal finances using the POCA (Proceeds of Crime Act 2002). This legislation reversed the burden of proof, meaning convicted criminals must show that they came by their funds legitimately or face them being confiscated. Levi (1997) describes how the idea of "taking the profit out of crime" has accelerated in Europe since the 1991 Council of Europe Convention on Money-Laundering and Confiscation of the Proceeds of Crime. By confiscating funds, you take away the incentive to commit an acquisitive crime and so disrupt criminal activity. The Home Office (2019) in its Asset Recovery Plan sets out its four objectives as:

- Disrupt criminal activity and the further funding of crime;
- Deprive people of their proceeds of crime;
- Discredit negative role models in society, and
- Deter people from becoming involved or continuing in crime.

Essex Police in their Proceeds of Crime Policy state "When a person has benefited from their crime, the Force's objectives are to secure a criminal conviction and, if possible, remove the benefit of that crime. A confiscation order made under POCA is an effective way of achieving this" (Essex Police 2018). Povey et al. (2004) suggests that 70% of all crime is acquisitive, giving the most quoted value of the proceeds of crime as 2% of UK GDPR or £18 billion. Data shows that less than 50% of the value of confiscation orders were recovered between 2003 and 2013. The data also shows that UK law enforcement are responsible for 39−59% of the total confiscated funds(Chistyakova et al., 2019).

Levi (1997) points out that POCA has the potential for making some areas of policing self-funding. Chistyakova et al. (2019) describes the Asset Recovery Incentivisation Scheme that allows UK law enforcement to keep a percentage of the recovered funds. Most Police forces now have a team of financial investigators dedicated to pursuing forfeiture and confiscation orders, and many UK forces have set up proactive money laundering teams. Kent Police state that the role of their proactive money laundering team is to "investigate organised crime groups and persons involved in money laundering and cash smuggling activities across Kent, including those operating across Kent's borders and further afield" (Kent Police n.d.).

A recent decision in AA v Persons Unknown & Ors [2019] EWHC 3665 (Comm) (13 December 2019) ruled that Bitcoin and other cryptocurrencies are property. Therefore, cryptocurrencies can be seized to prevent dissipation under section 47 of the Proceeds of Crime Act (POCA) 2002 (Home Office 2018). However, anecdotal evidence suggests that unless cryptocurrency assets are quickly seized, they are lost. This makes identifying the possession of cryptocurrency assets and finding the corresponding private key during the initial enforcement action, essential to the successful seizure of those assets.

1.2. Policing, crime and cryptocurrencies

The advent of The Onion Router (TOR) and Bitcoin gave rise to illegal dark web marketplaces, like *AlphaBay*, *Silk Road* and *Dream Market* (Weber and Kruisbergen 2019). These marketplaces catered for the sale of illegal goods and services, with payment being made in Bitcoin. Around the same time, ransomware started to evolve. While some forms of cyber extortion existed before Bitcoin, there were significant risks associated with obtaining payment and realising it through money laundering schemes. In 2008 the introduction of Bitcoin changed that.

From 2013 to the current time, almost all ransomware uses Bitcoin as the payment method (Richardson and North 2017; Hampton and Baig 2015). Hampton and Baig (2015) stated, "while some research states that ransomware may be easily defeated, but if history is a teacher, ransomware should not be dismissed as a passing fad". This statement has proven only too true. Richardson and North (2017) state that criminals have found that $10,000 is the optimal ransom, low enough that businesses will pay it and too small to attract the attention of law enforcement, whereas the FBI predicted that ransomware would become a billion-dollar business by 2016. The FBI had it right, given today's highly targeted ransomware attacks where demands are millions of dollars worth of Bitcoin. Colonial Pipeline has just paid around $4.4 million in Bitcoin for decryption keys (Gabbatt 2021).

However, ransomware and illegal marketplaces are not the only criminal enterprises to take advantage of Bitcoin. Bitcoin has become a popular asset for all forms of organised crime, including money laundering, extortion, child exploitation, drug trafficking, people trafficking and funding terrorism (Tziakouris 2018). Its popularity comes from the lack of money laundering regulations and controls that apply to fiat currencies and its pseudo-anonymous design that makes identifying end-users difficult. Foley et al. (2019) estimates that one-quarter of Bitcoin users and half of all Bitcoin transactions are involved in illegal activity.

1.3. Introduction to bitcoin

Bitcoin is the first of a growing number of cryptocurrencies in circulation and is still the most popular. Bitcoin allows peer-to-peer exchange of *coin* without a central financial institution. Bitcoin has value because people are willing to trade it for currency, goods and services. Bitcoin was introduced in a paper in 2008 by its creator, who goes by the pseudonym, Satoshi Nakamoto. It is based on cryptographically signed transactions stored on a distributed ledger, operated by a peer to peer network, using a proof-of-work scheme to avoid double-spend (Nakamoto 2008).

So if Alice wants to send a Bitcoin to Bob, first Bob must send his invoice address (based on Bob's public key) to Alice. Alice adds Bob's invoice address and the Bitcoin to transfer, to a transaction message, which Alice signs with her private key. Alice then transmits the transaction message onto the Bitcoin network. The Bitcoin miners gather transactions into blocks including Alice's, verify the transaction signatures and then compete with each other to produce a valid block using the proof-of-work algorithm. When a valid block is discovered it is appended to the Blockchain and the transactions are confirmed. Bob now has Alice's Bitcoin. The proof of work algorithm is designed to keep the miners finding a valid block approximately every 10 min.

1.4. Paper overview

Section II contains a literature review in the following areas: the underlying technology of Bitcoin wallets and address, the state-of-the-art in Bitcoin forensics, Policing and cryptocurrency assets and the Proceeds of Crime Act. These four research strands inform the design of a framework for Bitcoin triage which is discussed in Section III. The triage process is designed to be performed by law enforcement, during the lawful search of a property. This will be augmented with the guidance and best practices from the College of Policing (CoP) for digital evidence. In sections IV and V, the paper examines the practicality of recovering private-keys, through forensic methods or password attacks. As well as establish how any recovered private keys can be validated and then used to assist with the seizure of cryptocurrency assets.

2. Literature review

The literature review focuses on four main areas: live and offline digital forensics; how Bitcoin works; Bitcoin forensics; and wallet recovery. The research on offline and online digital forensics and Bitcoin will assist with the design of a framework for live Bitcoin forensics. To recover Bitcoin we have to understand how the keys are created and how they relate to addresses and transactions. The literature review will then explore the state-of-the-art in Bitcoin application forensics and wallet recovery. This will be restricted to Bitcoin forensics on a Windows host computer, and will not include forensic work focused on the Blockchain as this will not assist in recovering private keys. In looking at wallet recovery, it will examine popular attack methodologies and attack optimisations.

2.1. Live and offline digital forensics

Jafari and Satti (2015) examine a number of digital forensic models and find most models have the same four common processes. These are also the same general processes identified by NIST (Kent et al., 2006).

- Collection - Identifying and preserving information of interest.
- Examination - Processing of collected data to assess the information of particular interest.
- Analysis - Derivation of useful information that addresses the questions in issue.
- Reporting - Reporting the results of the analysis.

Traditional offline digital forensics involves examining a seized computer in a lab environment. The computer hard disk or solid-state disk is removed and forensically imaged. The image is then examined using forensic tools to recover information that may assist the investigation (Rafique and Khan 2013). There are several advantages and disadvantages to this approach. The main advantage is that this is a tried and tested forensic method which prevents changes to the data under examination. This means that the process is highly repeatable. Although garbage collection and wear levelling technologies in Solid State Drives (SSD) cause repeatably issues (Bell and Boddington 2010). The most significant disadvantages are that encryption can prevent data recovery and any volatile data; for example, the content of Random Access Memory (RAM) is lost. The RAM may have contained encryption keys, evidence of file-less malware, passwords or evidence of the offence under investigation.

Live forensics is the process of collecting data from a running computer and involves interacting with the 'live' system to run forensic recovery tools or use tools built into the Operating System (OS) to recover information, often called *living off the land*. This approach will alter the data on the computer to some extent, which is the biggest disadvantage. Carrier (2006) also points out the risk that any tools built into the operating system could have been altered to provide unreliable results. Another issue is repeatability. Forensic tasks like RAM capture are unrepeatable by their very nature (Case and Richard, 2017). Non-repeatability makes it difficult to validate that the method being used is performing as expected. It also prevents a 3rd party forensic expert from fully validating the results(Zhang et al., 2015). Zhang et al. (2015) describe the difficulties of assessing how forensically sound their GPU memory recovery technique is. However, live forensics is becoming critical due to the default use of encryption and the growing use of cloud services.

Live forensics is often quicker to perform as it targets very specific artefacts rather than imaging the entire disk. A digital forensic practitioner who examines a live device needs to be acutely aware of the consequence of every action taken on the device under examination and weigh up the advantages over the possible loss or contamination of the evidence. Due to the complexities of modern operating systems, this is not an easy task. With lots of applications turning to cloud solutions, live forensics becomes far more important, as cloud data may not reside on the device being examined. In the UK, legislation under section 20 of the Police And Criminal Evidence Act 1984 (PACE) can be used to capture cloud-based data during the execution of a search warrant or other statutory search power, where the data is accessible from the premises.

All digital forensic work conducted by UK law enforcement must follow the ACPO four principles of digital evidence (Williams 2012). Principle One states that no action taken by law enforcement should change data that is subsequently relied upon in court. Principle Two allows accessing live data where it is necessary, but the operator must understand the relevance and impact of their actions. Principle Three requires a complete audit trail of any operations conducted. Principle Four makes the person in charge of the investigation responsible for compliance with the principles.

Therefore, any forensic model or tool for use by UK law enforcement must be designed with these principles in mind. How does this affect live forensics techniques? We have already identified that anything done live on a device will alter data. The answer is it depends on the circumstance. Kent Police developed the mnemonic JAPAN to help officers with making decisions in compliance with the Human Rights Act. JAPAN stands for Justified, Authorised, Proportionate, Auditable and Necessary (Akhgar and Wells 2018). This can be applied to the question of performing live forensics. Live forensic methods should only be used where these questions can be answered positively concerning the aim of the investigation. If not, an offline forensic approach should be used instead. We define two general scenarios where the JAPAN test would likely be met during the execution of a warrant.

- Where the data would otherwise be lost to law enforcement. For example, where encryption is in use, or cloud storage.
- Where the information is required for a pressing operational need. For example, where information is required urgently to protect life, prevent serious harm, or to prevent the loss of evidence in a serious crime.

We argue that the live forensic examination of devices to assist the seizure of cryptocurrency assets would meet the JAPAN test for the following reasons. It would be justified as it fulfils the aim of reducing crime by *taking the profit out of crime* under POCA. It would be necessary because if the cryptocurrency asset is not seized as soon as practicable, there is a significant risk that it will be lost. It would be authorised under section 47 of POCA 2002. Proportionality would be based on the offence under investigation, this is likely to be proportionate for any serious crime. It would be auditable as the recovery would follow ACPO Principle three.

2.2. Bitcoin

2.2.1. Hardware wallets

Hardware wallets are considered one of the more secure ways to store cryptocurrency private keys, since the private keys never leave the custom hardware. The popular Ledger hardware wallet is shown in Fig. 1. Hardware wallets almost universally use a PIN to secure the device, some have now started to offer biometric authentication. There has been some research on side-channel attacks against hardware wallet PINS (San Pedro et al., 2019). However, these rely on device-specific vulnerabilities which are quickly

patched once disclosed. They also require special equipment, so they are unlikely to be practicable for a triage process.

However, these hardware wallets still need software running on a computer to allow transaction signing. This is part of the design, the separation of the cryptographic keys from the rest of the system. Thomas et al. (2020) use memory forensic techniques to extract artefacts from the windows Ledger client using their FORESHADOW volatility module. They found that while the Ledger client application was running, several extended public keys were identifiable. However, 6 min after the application is terminated all traces had been overwritten. Thomas et al. (2020) also found that extended public keys could be recovered from the Trezor application.

One thing that is common for all hardware wallets is they force the user to record the recovery seed words, before making the user reenter them. For law enforcement, finding a hardware wallet should encourage a more detailed search for the recovery seed words. Anecdotally, recovery seeds are often found in the empty hardware wallet box or written in notebooks.

2.2.2. Software wallets

There are a large number of Software wallets for Windows, MacOS, Android and iOS. This paper will concentrate on wallet software for Windows. The popular Electrum wallet is shown in Fig. 2. Software wallets come in two categories: full clients which download and store the entire Blockchain; and light-weight clients that leverage an online API to function (Van Der Horst et al., 2017). Most applications fall into the latter category as it is undesirable to have to download and store the entire Blockchain which is currently about 370 Gigabytes and increasing all the time (blockchain.com, 2021). One of the most popular Windows software wallet applications is Electrum a light-weight client (Van Der Horst et al., 2017; Zollner et al., 2019). Unlike a hardware wallet, to be able to sign transactions, the software wallet must at some point load the wallets private keys in RAM. These private keys must be stored or the user would have to enter them every time they wished to conduct a transaction. Typically, they are encrypted with a user-supplied password and stored on disk. Both of these design constraints provide possible attack vectors.

2.2.3. Hosted wallets

Coinbase is one of the larger cryptocurrency exchanges; they offer the option of hosted wallets. A hosted wallet is where the provider holds the private keys. This is similar to a bank, and has the advantage of you not being solely responsible for your keys, which if you lose, you are unlikely to ever regain access to the cryptocurrency in the wallet. The downside of hosted wallets is the 3rd party controls your keys, so you have to trust that they will do so securely. For law enforcement, this means that it is likely that Coinbase can freeze assets with the appropriate legal process. Coinbase's FAQ states "In extremely rare circumstances, and only where required by law, Coinbase may block or freeze customer funds on our platform. We will take this action only when we are required to comply with an order from a court or other authority that has jurisdiction over Coinbase, which compels us to restrict access to funds." (Coinbase n.d.).

2.2.4. Blockchain API

A Blockchain Application Programming Interface (API) is a mechanism developers can use to query the Blockchain from their applications, without having to download the entire Blockchain or developing their own API. There are a large number of commercial Blockchain API's which have free and paid for services (Parlika and Pratama 2021). For example, using a Blockchain API you can retrieve the current balance of a Bitcoin address and all transactions associated with it.

2.2.5. Key generation

Bitcoin uses Elliptic Curve cryptography with the SECP256K1 curve. The parameters for this curve are documented in Certicom Research's SEC 2: Recommended Elliptic Curve Domain Parameters (Brown 2010). Bitcoin uses 256-bit private keys, which need to be uniformly random in the range 1 to n-1, where n is the curve order defined by the SECP256K1 parameters. Public keys are then generated from the private key using Elliptic Curve Digital Signature Algorithm (ECDSA). Private keys are often stored in Wallet Import Format (WIF). Public keys are often stored in a Base58Check encoded form. Public keys can be uncompressed or compressed using the Standards for Efficient Cryptography (SEC) format. Key pair generation where Q is the public key, d is the private key, n is the curve order and G the base point, can be expressed as: Select $d \in_R [1, n-1]$, Compute $Q = dG$, Return (Q, d) (Courtois et al., 2016). Python code to generate a key pair is shown in Fig. 3.

Private keys are stored in WIF which contains a header indicating if the key is for the mainnet (0x80) or testnet (0xef). Then

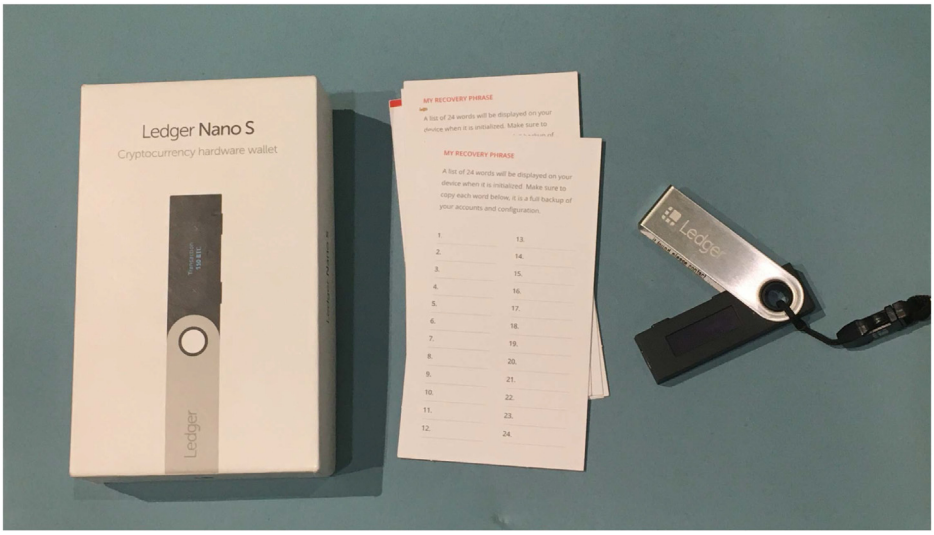

Fig. 1. Ledger nano S hardware wallet.

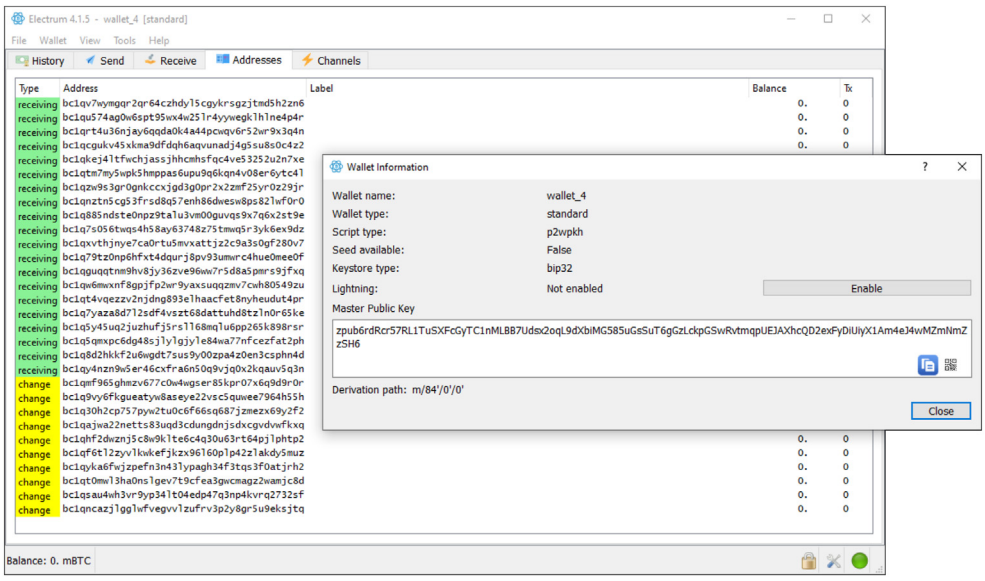

Fig. 2. Electrum software wallet.

comes the 32-bit private key, optionally followed by a byte containing '0x01' if the corresponding public key is compressed. It is all then Base58Check encoded. The Base58Check checksum is generated by taking the first four bytes of the result of SHA256 hashing the data twice.

Public keys can be compressed using *y*-coordinate point compression. An elliptic curve public key is an *x*, *y* coordinate on the elliptic curve. A public key is compressed by removing the *y*-coordinate and pre-pending a positive (0x02) or negative (0x03) indicator. From this, the y coordinate can be recovered later as the equation for the curve is known. This almost halves the size of the public key. Public keys are then Base58Check encoded.

2.2.6. Invoice addresses

There are currently three types of invoice addresses in use on the Bitcoin network. Pay to Public-key Hash (P2PKH) which begin with the number one. Pay to script hash (P2SH) which begin with a three are often used for multisig addresses. Finally, native Segwit addresses (Bech32) the newest address type which addresses start with 'bc1′ for the main net and 'tc1' for the testnet.

P2PKH addresses are calculated from the public key by hashing it with SHA256 and then RIPEMD-160, the result is then prepended with 0x00 for mainnet and 0x6f for testnet. Again the result is Base58Check encoded. A Python example is shown in Fig. 4.

2.2.7. BIP32 deterministic wallets

Bitcoin Improvement Proposal (BIP) 32 describes a scheme for deterministic wallets. A deterministic wallet is one where a hierarchical set of Bitcoin addresses and keys can be generated from a single high entropy random 128-bit to 512-bit master seed (Wuille 2012). From this master seed, you generate a private root key *m* as follows HMAC-SHA512(key = "Bitcoin seed", data = masterSeed). The private root key is then used to generate a tree of private child keys using the CKDpub function described in BIP32 (Wuille 2012). Fig. 5 shows the structure of a BIP32 deterministic wallet.

```
import secrets
import ecdsa
def generateKeyPair():
    rnd = secrets.randbits(256)
    privKey = rnd % ecdsa.SECP256k1.order-1
    privKey = privKey.to_bytes(32,'big')
    pubKey = ecdsa.SigningKey.from_string(
        privKey, curve=ecdsa.SECP256k1).verifying_key
    pubKey = pubKey.to_string()
    return([pubKey,privKey])
```

Fig. 3. A python example of bitcoin key generation.

2.2.8. BIP39 recovery seeds

The BIP39 sets out a method of using a mnemonic code to record a master seed for generating deterministic keys. The motivation was to find a human-compatible way to record a computer-generated random seed. The mnemonic sentence is made up of a carefully chosen dictionary of 2048 words. The words are chosen to avoid similar words and can be uniquely identified by the first four characters (Palatinus et al., 2013). To generate a mnemonic code, first *ENT* a high entropy random number is generated. *ENT* can be 128, 160, 192, 224 or 256-bits long. The number of checksum bits is calculated with $l = len(ENT)/32$. A checksum *cs* is generated using the first *l* bits of a SHA256 hash of *ENT*. This gives a seed value of *ENT*+*cs*. The number of mnemonic words that will be generated can be calculated by $(len(ENT) + l)/11$. The seed value is split up into 11-bit values. Each 11-bit value specifies an index into the dictionary. So for a 128-bit seed, the total size of the seed will be $(128 + 4)/11 = 12$ mnemonic words. There are different dictionaries for different languages. *ENT* is not used as the seed for generating keys. An optional password can be added to the end of the mnemonic words, if no password is selected, an empty string is used. The master seed is generated from the seed words and a salt which is the word "mnemonic" concatenated with the password. These are used with a PBKDF2 function with 2048 rounds and HMAC-SHA512 as the pseudo-random function. The result is a 512-bit (64 byte) key. From this 512-bit key, the extended private (xprv) key is

```
address = b"\x00" + ripemd160(sha256(pubKey))
checksum = sha256(sha256(address))[:4]
encodedAddress = base58(address + checksum)
```

Fig. 4. A python example of bitcoin address generation.

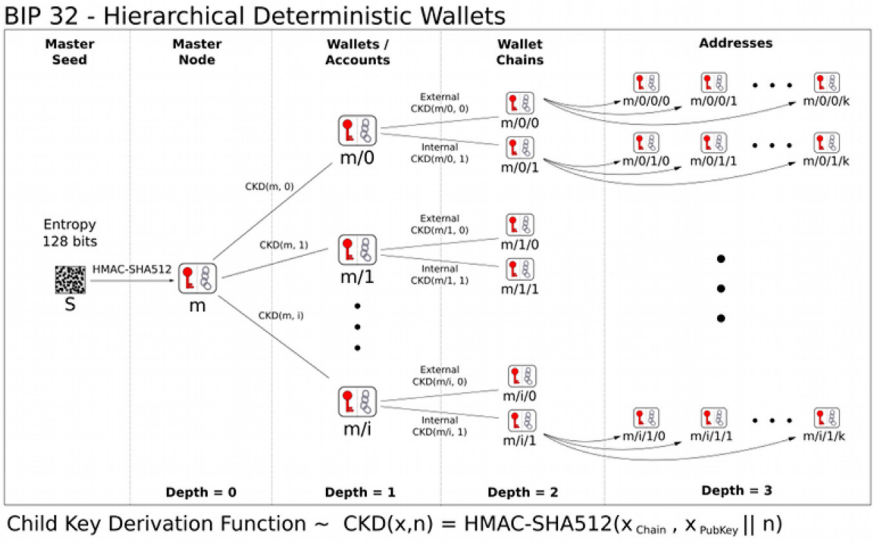

Fig. 5. BIP32 - hierarchical deterministic wallets (Wuille 2012).

generated in the same way as described in BIP32 (Palatinus et al., 2013; Rusnak 2013). Python code to generate the master seed from a BIP39 recovery seed is shown in Fig. 6.

2.3. Bitcoin forensics

There appears to be limited academic research in this area, what research there is comes from a small group of academics in conjunction with industry and law enforcement. There are a large number of Bitcoin applications, especially on iOS and Android. To date, there has been limited research on the security of these applications (Van Der Horst et al., 2017). This review will concentrate on Bitcoin wallet applications running on the Windows platform.

Van Der Horst et al. (2017) examine the forensic artefacts created by Bitcoin core and Electrum, two of the more popular desktop wallet applications. They concentrate on RAM artefacts but also point out most of the artefacts they found can also be located on the disk. They use virtual machines, on which they install and operate the wallet software under test. Then examine the virtual machines memory file when the virtual machine has been suspended. This approach has several advantages. It does not require 3rd party software to dump the content of RAM and avoids the issue of RAM smearing, which occurs due to ongoing changes to RAM during the capture process (Case and Richard, 2017).

```
import hmac
import hashlib
mnemonic = "rude original bachelor leave
            round toss lend awful
            behave elite april super"
passphrase = ""
passphrase = "mnemonic" + passphrase
mnemonicBytes = mnemonic.encode("utf-8")
passphraseBytes = passphrase.encode("utf-8")
masterSeed = hashlib.pbkdf2_hmac("sha512",
                          mnemonicBytes,
                          passphraseBytes,2048)
print("Master seed:",masterSeed.hex())
```

Fig. 6. A python example of BIP39 master seed generation.

Of greatest interest are Van Der Horst et al. (2017) results and their ability to recover private keys or seed words. They conclude that they can only recover the private keys from Bitcoin Core when the wallet is unencrypted. They were unable to recover the private keys from Electrum. They were able to recover the seed words from Electrum but only immediately after wallet initialisation. Van Der Horst et al. (2017) acknowledge that the private keys they recover from memory are also recoverable from the wallet file on disk. Capturing and processing RAM is significantly more time consuming than just parsing the unencrypted wallet file and so offers no advantages for private key recovery in this case.

Zollner et al. (2019) combine live and postmortem forensic to analyse Bitcoin artefacts on Windows 7 and 10. Their stated aim is to locate and extract important Bitcoin artefacts, such as Bitcoin keys and addresses. They go on to identify that this information may be used for asset forfeiture in proceeds of crime legislation. Zollner et al. (2019) present an automated Bitcoin analysis tool WinBAS. While their paper describes it as open-source, only a compiled executable is available in the indicted GitHub repository. They identify that it is trivial to identify Base58Check encoded keys as they only contain symbols in the Base58 alphabet and the checksum verifies that the recovered key is valid. The same is true of extended private keys due to the prefix 'xprv'. However, they exclude looking for keys in hex decimal format due to overwhelming false positives. Zollner et al. (2019) also suggest looking for prefetch files associated with Bitcoin wallet executables. This is a fast approach to identify which applications have been used, but is dependent on prefetch being enabled. In their paper, they mention that encrypted wallet files may be recovered and that one approach would be a brute force attack. However, they do not examine this further.

2.4. Wallet recovery

Wallet recovery aims to enable the seizure of any available crypto-assets. This can only be done if the collection phase has recovered the wallet's private key. However, for security most wallet applications encrypt the private key with a user-supplied passphrase. In the case of a hardware wallet, the key is inaccessible as it never leaves the hardware device. On top of this, hardware wallets are usually secured by a PIN to prevent unauthorised

use (Ledger n.d.). If the recovery seed words for a hardware wallet are found they can also be protected with a user-supplied passphrase.

If an encrypted wallet file is located, attempts to guess the password can be made. The success of a password guessing attack is based on the key derivation function used and the strength of the password. Several open-source tools can perform these attacks, including Hashcat and BTCrecover. Both BTCrecover and Hashcat support attacks on MultiBit and Electrum wallets, amongst others(Hashcat 2021; Rothery 2021).

If a hardware wallet recovery seed is found you can try to guess the associated passphrase. The logical first step is to try an empty string, this is the case where no password has been set. From the recovery seed and password, a master seed is generated using the PBKDF2 function. The wallet extended public key must then be calculated, which requires an elliptic curve point multiplication. If an extended public key has been recovered, it can be compared with the calculated extended public key to ascertain if the password was correct. If not a set of wallet addresses would have to be calculated again using elliptic curve point multiplication. However, every passphrase will result in a valid set of wallet addresses. So the Blockchain needs to be interrogated to see if any of the calculated addresses have appeared on the Blockchain. If any of the addresses have appeared on the Blockchain the guessed password is highly likely to be correct.

Courtois et al. (2016) examines speed optimisations for elliptic curve point multiplications. They benchmark existing implementations and show they achieve a 2.5 times speed increase at a cost of more RAM by pre-computing data that only depends on the curve.

2.4.1. Password attacks

Knowledge of how people choose passwords can be used to increase the chance of guessing the right password. With your cryptocurrency wallet forgetting the password could lead to the total loss of any cryptocurrency assets stored in the wallet and therefore, the password may be recorded somewhere or designed to be memorable. There has been considerable research conducted on how people choose passwords and how secure any specific password is. A naive approach to calculating password strength is to use the Shannon Entropy metric for information (Pearman et al., 2017). This is $E = log_2(R^L)$. Where E is the password entropy, R is the size of the pool of unique symbols, L is the number of symbols in a password and R^L is the total number of combinations possible. The value E is the lowest number of bits required to represent the number of possible password combinations (Shannon 1948). However, this does provide a way of representing the number of possible combinations that can be directly compared with the size of encryption keys. Table 1 shows the calculated entropy of different password schema, if the password is chosen at random from the set described by the schema. These are many orders of magnitude smaller than standard encryption keys, for example, a 128-bit AES key. This makes guessing the password easier than guessing the encryption key.

To guess passwords there are two popular approaches. First, a brute force attack. A brute-force attack tries every possible combination. For example, brute-forcing a four-digit PIN you would start at 0000 then try 0001 up to 9999. Second, a dictionary attack, where you compile a dictionary of more likely passwords and make guesses from this list starting with the most likely. Let us examine a brute force attack. If the PIN was chosen at random, on average you would have to try half the password space to guess the password, so 500 guesses. If you could guess 10 PINs a second on average it would take 50 s to correctly guess the PIN, and at most 1 min 20 s. However, if you could only make one guess every minute, it would take just over 8 h on average. Slowing down guessing attempts is the approach most password key derivation functions use to prevent successful attacks. This is done by making the key derivation function take enough time to make attacks impractical but fast enough not to be noticeable to a user, this is key-stretching. Depending on the implementation of the key-stretching algorithm, it can be overcome to some extent using custom hardware, a Graphics Processing Unit (GPU) or distributed computing. Brute force attacks become less practicable if we start looking at 12 character passwords chosen from upper and lower case characters, the digits 0 to 9 and special characters, due to the number of possible combinations. Therefore, success is more likely with a dictionary attack, taking advantage of the weaknesses in the way people choose passwords. A simple approach to building a dictionary is to gather leaked password data sets. Then analyse the data set for the frequency each password appears in the data set, and sort them with the most frequent passwords first.

There are two ways to make your dictionary attack more effective, guess faster and guess smarter using a better dictionary. Guessing faster is often constrained by current hardware and can be costly to implement. Currently, an NVIDIA RTX 3090 based graphics card retails for around £2000 (Scan Computers International Ltd 2021). Guessing smarter was explored by Weir et al. (2009) who examine large data sets of leaked passwords to calculate the probability of specific password structures occurring. They compared their approach to the popular John The Ripper application using several dictionaries. They found that their system cracked 28%–129% more passwords.

Wang et al. (2018) conducted a study of large data sets of password breach data from real-world sources. They identified that 34.3% of users used the same password across multiple services and a further 12.1% used the same password modified with a simple rule. Pearman et al. (2017) also found high password reuse finding that users in their study used 9.88 passwords across 26.34 websites. If a triage tool can extract users passwords for other services, the research suggests that they can then be used effectively, to inform password guesses.

Table 1
Entropy for specific password schema.

Password Schema	Entropy (bits)
8 characters [a-zA-Z0-9]	48
8 characters [a-z]	38
3 random words from BIP39 English dictionary	33
1 random word from top 10,000 English words	14
1 random word from BIP39 English dictionary	11

2.4.2. Password based key derivation functions

Password-Based Key Derivation Function Two (PBKDF2) is a popular password-based key derivation function. PBKDF2 uses a Central Processing Unit (CPU) intensive pseudo-random function with a large number of iterations to make it slow to compute (Visconti et al., 2015). PBKDF2 is used by BIP39 to calculate the master seed from the recovery seed words and passphrase using 2048 iterations of HMAC-SHA512. However, while PBKDF2 is CPU-intensive it requires minimal memory to compute and is trivial to compute in parallel. Therefore, it can be accelerated using a GPU or an Application Specific Integrated Circuit (ASIC). These typically contain a large number of small processing units that can access a large global memory, a small amount of fast local memory and a limited number of registers (Ruddick and Yan 2016). There are two primary programming languages used to write GPU code, Compute Unified Device Architecture (CUDA) which is NVIDIA specific and Open Computing Language (OpenCL) which is supported by a large

range of GPUs. OpenCL and CUDA GPU code can be leveraged in the Python programming language with the appropriate Python modules (Holm et al., 2020).

Visconti et al. (2019) examined the practicality of brute-forcing passwords stretched with PBKDF2 for full volume encryption on Linux. Rather than just looking at the number of passwords per second that can be processed, they take a different approach. They compare the dollar cost of the password attack. In their calculation, they included the cost of the hardware, its life expectancy and the cost of the electricity used. The disadvantage of this approach is that their results will become outdated very quickly as costs change and hardware performance increases. This is minimised by only comparing the relative difference in cost between the hash functions and platforms examined. Visconti et al. (2019) also show that using consumer-grade (Nvidia GTX/RTX series) GPUs rather than commercial-grade (Nvidia Tesla series) GPUs to conduct password attacks is more cost-effective. An approach adopted by this paper.

Ruddick and Yan (2016) explore why oclHashcat outperforms competitors calculating many cryptographic primitives, including PBKDF2; before proposing their own optimised algorithm to calculate PBKDF2 using OpenCL running on cheap consumer-grade GPUs.

Benchmarks conducted with Hashcat running on an Nvidia RTX 3090 GPU show it can calculate 1,232,000 PBKDF2-HMAC-SHA512 operations using 999 iterations in 1 s. The Hashcat benchmark outputs are shown in the appendix. This suggests a maximum hash rate of about 600,000 hashes per second, with 2048 iterations. There are functions like scrypt that are memory expensive and so cannot be successfully accelerated with GPUs (Visconti et al., 2019).

2.4.3. Cloud compute platforms

There are now many different cloud-based compute instances that can be rented for a short to a long duration. These are offered by the big three cloud providers Amazon, Google and Microsoft as well as a range of mid-sized and small vendors, like OVH and vast.ai. Each of these vendors has multi-GPU instances that have per second/minute billing plans and an API so that provisioning and destroying instances can be scripted. These cloud compute platforms allow you to provision a large number of GPUs for a short period. This makes them ideal for providing the compute resources necessary to conduct short-duration password attacks economically. There is an almost linear relationship between the number of GPUs employed and the number of passwords per second that can be processed. Therefore, the number of passwords you can guess is only limited by the number of compute instances you can instantiate. If you double the number of GPUs you will double the cost but almost double the number of passwords per second that can be guessed. For a specific attack, the number of password guesses per unit cost can be calculated. Amazon, Google and Microsoft employ costly commercial GPUs like the NVIDIA Tesla V100. Vast.ai is a platform where individuals can sell time on their GPU platforms. These are generally lower cost consumer-grade GPUs. Fig. 7 shows a typical current price for an instance with a single NVIDIA RTX 3090 GPU.

2.4.4. Wallet password attacks

Volety et al. (2019) examine cracking Electrum and MultiBit wallet recovery seeds on a Windows desktop environment. MultiBit wallet was depreciated in 2017 (Heutmaker 2017). The process they use is to extract the seed word dictionary from the wallet's running process. They then create a database holding all possible combinations of 12 dictionary words. Using automation software they automate the recovery process using the seed words and the wallet application. They find that between 1% and 10% of the combinations entered are accepted by the application. They state that this method is very slow compared with other offline brute force methods. When they find a set of accepted seed words, they have control over the wallet, but if the address has never been used they find an empty wallet. The paper gives no idea of the time per guess using their method, or the number of empty wallets to active wallets discovered. This approach is inefficient as most of the guesses are rejected by the software because the BIP39 checksum is invalid. These could have been removed before being added to the database. From the BIP 39 specification, we know the number of valid possible combinations is 2^{128}. Therefore, the probability of finding a wallet that is in use is very small. Chainalysis (2018) suggest that there are 640 million addresses in use on the Bitcoin Blockchain; the number of wallets in use will always be less than this as a wallet can hold multiple addresses. We can approximate the chance of finding an wallet in use with $\frac{3.4\times10^{38}}{6.4\times10^{8}} = 1$ in 5.3×10^{29}. Therefore, this is not a realistic approach.

Vasek et al. (2016) conducts a dictionary attack on brain wallets using their open-source application BrainFlayer. Brain wallets generate the master seed from a password or phrase. They tried over 300 billion passwords from a custom set of dictionaries. From this, they only identified about 1000 addresses that have previous transactions. One of the reasons they were able to guess so many passwords was that they were only using a single SHA256 hash to generate the master seed from the password. However, they still needed to identify addresses in use for each guess. Vasek et al. (2016) describe how they use an optimised secp256k1 library from Bitcoin core to calculate the associated invoice address. Then having downloaded the entire Blockchain extracted all the invoice addresses using the znort987 block passer and loaded them into a sorted bloom filter, they were able to quickly identify addresses that may have been used. If they found an address possibly in use, they made a request to the Blockchain.info API for all the details and current balance. This work is very relevant to this paper, and the approach taken by Vasek et al. (2016) has some advantages with one disadvantage. The advantages are that each address lookup in the bloom filter is very quick. The lookups will be orders of magnitude quicker than an API call to an online resource. The only disadvantage is that the bloom filter requires continual updating and will always be *out of date*.

3. A framework for bitcoin triage and asset recovery

This section examines the results of the literature review to propose a framework for Bitcoin triage. The framework will adhere to the popular four-stage forensic model. The framework will also take into account guidance from the Association of Chief Police Officers (ACPO) digital principles for digital evidence (Williams 2012). Following Principle One and Two means we must minimise the tool's footprint on the target machine. It is also necessary to hash the data obtained to ensure its ongoing integrity. Lastly, to comply with Principle Three, the tool needs to log all the actions performed.

This is by design a triage tool to recover Bitcoin private keys. The intention is not to acquire all possible forensic artefacts for evidential purposes as this is better performed later in a laboratory setting. As this is a triage process, it should be something that an investigator can perform during a premises search. It is unlikely to be practicable if the process from beginning to end takes longer than 2 h. It should also be heavily automated, so it can be operated by a relatively unskilled technician. An overview of the process is shown in Fig. 8.

The collection phase will take place on the live target system and extract relevant artefacts which will be saved to a removable storage device. The collection phase can be augmented with the

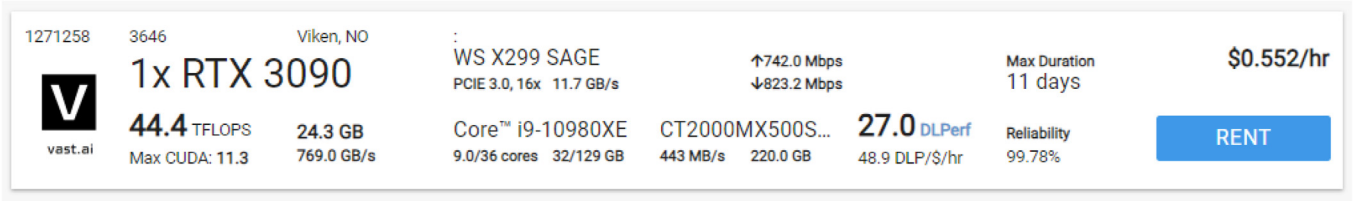

Fig. 7. Vast.ai RTX 3090 instance pricing.

results of any physical search. This may include a recovery seed or passwords that had been written down. The examination phase will be conducted on the examiner's laptop computer at the scene. It will extract wallet types, keys, addresses and passwords from the recovered artefacts. The analysis phase will be performed on a cloud compute platform, this is because typically the examiner's laptop will not have sufficient compute performance to perform wallet attacks in a reasonable period. This phase will identify wallets that can be attacked and then perform the attack. This would include dictionary attacks against encrypted wallet files or recovery seeds, identifying unencrypted wallets and wallets of significant value. By offloading this to the cloud, it will allow massive parallelisation to increase the number of attempts at guessing the password per second.

3.1. Collection

The aim of the collection phase is to gather relevant artefacts (Kent et al., 2006) including:

- Artefacts that indicate that Bitcoin is in use
- Artefacts that contain private keys
- Artefacts that contain recovery seeds.
- Artefacts that may assist the recovery of private keys.
  - Artefacts that contain public keys
  - Artefacts that contain passwords or passphrases.

To reduce the collection processes overall footprint, the application should have a relatively small memory footprint. It should also not change data on the target computer unless it cannot be prevented. If changes are made, they should be documented in the logs. An overview of the collection phase is shown in Fig. 9.

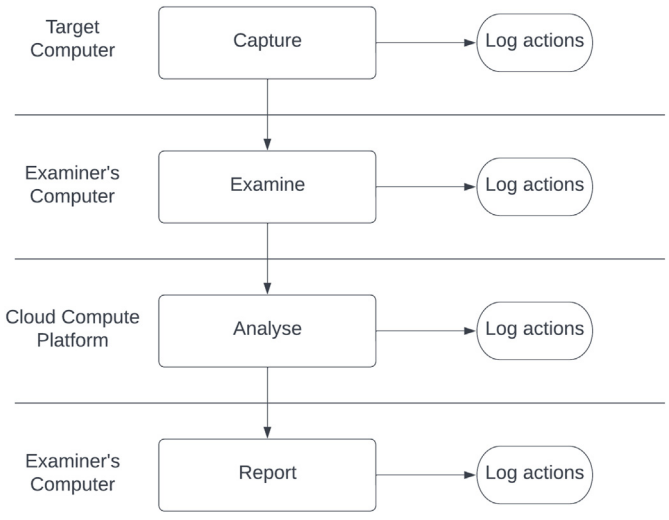

Fig. 8. Framework overview.

3.1.1. Prefetch files, registry and known file locations

These searches aim to quickly identify wallet applications installed or used on the target system. Searching for registry keys relating to specific applications or the default installation directories can identify the presence of these applications. Zollner et al. (2019) show prefetch files are useful to identify the wallet applications and browsers in use on a system. While this requires prefetch to be enabled, it is enabled by default on desktop Windows 10 builds. However, access to the prefetch files requires Administrator privileges, so it may not always be possible. Prefetch files are stored in the folder "C:\windows\prefetch". There are several utilities for parsing these files, including Nirsoft's WinPrefetchView (Sofer n.d.). Identifying specific wallets applications that have been run or installed on the system allows us to target specific known application artefacts. Including registry keys, configuration files, log files and wallet files which can be captured very quickly.

3.1.2. Running processes

Van Der Horst et al. (2017) investigates Bitcoin Core and Electrum wallet forensics, focusing on the analysis of the applications process memory. This is normally done by capturing the whole of RAM and then extracting artefacts with string searches or the volatility framework (The volatility Foundation, 2020). It is also possible to dump only a specific processes memory; a popular utility for this is ProcDump a Microsoft Sysinternals tool (Russinovich and Richards 2021). Dumping just the process you are interested in is faster, and the data is limited to the specific application of interest. In contrast, dumping the entire contents of RAM is slower, but may allow the recovery of artefacts from terminated processes and closed files. Anecdotal reports suggest that the process of RAM capture can sometimes cause Windows to crash. This must be considered, as we may lose access to volatile data on the system due to a crash.

A software wallet cannot operate without having the private keys in RAM at some point during the signing of a transaction. Van Der Horst et al. (2017) find that they recovered a large number of public keys and, in limited cases, private keys from process memory, demonstrating how useful this artefact is.

3.1.3. Browser artefacts

Several useful browser artefacts can be recovered, including browsing history and cached credentials. Browsing history may identify the use of a hosted wallet for example with Coinbase; in this case, legal routes may be available to freeze the assets directly through the hosted wallet provider (Coinbase n.d.). Recovered cached browser credentials can be added to any dictionary used to guess the password or passphrase. If several sets of credentials are obtained they may identify the schema used to generate the

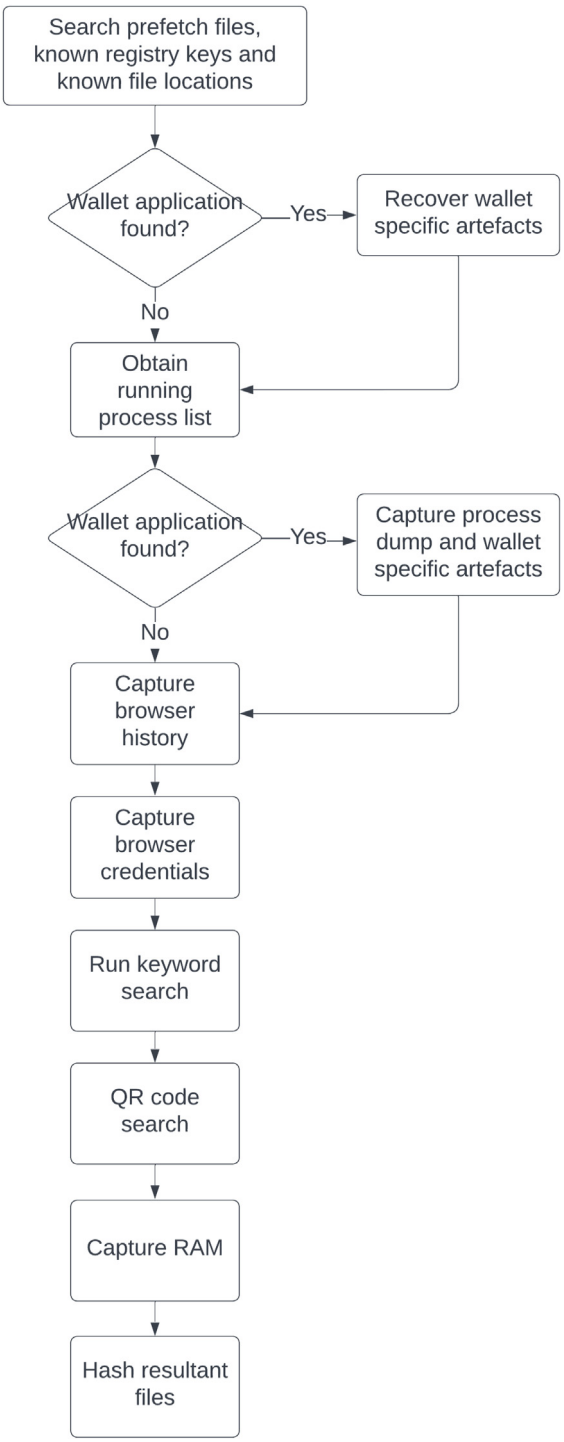

Fig. 9. Collection process.

passwords. This will be invaluable if a wallet file or recovery seed needs to be attacked as passwords can be generated using this schema which can be added to the password dictionary used in the attack. Sofer (n.d.) has written utilities that will extract both of these artefacts from the most popular browsers. However, the utility to extract cached browser passwords is often incorrectly detected as malware as it accesses sensitive information. This requires any real-time antivirus protection to be disabled before running the utility.

3.1.4. File searches

Keyword and regular expression searches of files, partitions and disks are slow compared to other operations. This type of low-level search will find artefacts in unallocated and slack space, but is limited by the read speed of the physical storage device. With newer computers having mid-sized fast SSDs it is more practical to perform keyword searches now than in the past. However, searching the disk from beginning to end is still unlikely to be practical for a triage operation. Bayne et al. (2018) explored the use of a Graphics Processing Unit (GPU) to speed up pattern matching, but since the collection phase is performed on the target system, we are limited by its hardware which may not include a GPU. However, practical searches can be achieved on most hardware by reducing the search space. This can be accomplished by only searching files in specific locations or specific file types and ignoring known operating system files. Keyword searches often use Regular Expressions to describe complex search patterns. Global Regular Expression Print (GREP) being the most popular tool (Forte 2004). Zollner et al. (2019) provides Regular Expressions for Bitcoin keys and addresses in several formats. Searching for saved recovery seeds could also be performed by searching for a run of 12–24 words from the 2048 word dictionary. Another option for a quick keyword search is to search the Windows file index database. Windows by default indexes certain file types and allows you to perform keyword searches against all indexed files in a matter of seconds (Chivers and Hargreaves 2011). However, this is very dependent on the search and file index setting on the target computer and will not work if the user has disabled file indexing. Regular expressions are also not supported.

3.1.5. QR code search

QR codes are convenient for exchanging Bitcoin addresses, most wallet software produces QR codes of keys and addresses. Image analysis is a relatively slow process. However, search speed can be increased by pre-filtering. It is unlikely any QR code saved as an image will be larger than a few tens of kilobytes and are predominantly black and white. The filtered images could then be analysed by QR code library, for example, ZBar (Brown 2011). After the data encoded in the QR code is recovered, it can be compared with popular key and address formats. Any matching artefacts can be saved for later analysis.

3.1.6. RAM dump

The RAM dump may allow the recovery of data from applications that have since been closed. However, this data is very ephemeral, and a large number of factors like usage and the time since the application was closed can affect successful recovery. User passwords may be contained within RAM and recovered by extracting all strings from the RAM dump. We need to extract both UTF-8 and UTF-16 strings. These strings can be added to a custom dictionary for later password attacks.

3.2. Examination

The examination phase is responsible for extracting data from the artefacts obtained during the collection phase which may assist us to identify a wallet's current value or help attack the wallet. This process will run on the examiner's laptop. The process starts by identifying any recovered wallets. If the recovered wallet is unencrypted the private keys are extracted. Where the wallet is encrypted xpub, ypub and zpub extended public keys are recovered from the artefacts where possible. These will allow the balance of the wallet to be obtained through a Blockchain API call. The first source of these are configuration files, then the wallet process dump. These sources tie the recovered keys to a specific wallet

application. Next, the RAM dump and other recovered files are examined, these sources are secondary as they may not be able to be tied to a specific wallet application. If the extended public key cannot be obtained, addresses associated with the wallet application should be extracted where possible. From a single address, it is not possible to calculate any other addresses that are part of a deterministic wallet. Therefore, any API calls for balance will only relate to that specific address, not the wallet as a whole. The process is summarised in Fig. 10.

Next, any recovery seeds are examined to determine if the wallet type can be identified, as not all wallets use BIP39 seeds. If not the recovery seed is treated as a BIP39 seed. From this the private key and extended public keys are calculated. These will allow the balance of a specific wallet to be queried through a Blockchain API call. The process is summarised by Fig. 11.

Identification of hosted wallets is performed by searching the browser history for known hosted wallet URLs. These will just be reported to the investigator. Brute forcing a hosted wallet, requiring an online attack, is not usually practicable due to the security measures put in place to prevent this. In some cases, it is unnecessary as the cryptocurrency assets can be preserved through legal requests to the service provider.

The next step is to examine the cached browser passwords. Users practise poor password security and research has shown password reuse is common (Wang et al., 2018; Pearman et al., 2017). Therefore, this may identify passwords for wallets.

If any wallets or seeds have been found then as part of the examination phase, a list of passwords is generated. These would be generated from the browser password cache and may be expanded with password mangling rules. Strings can also be extracted from the RAM dump which can be added to the password list. Adding actual users passwords to any custom dictionary later used to attack wallets or seed words is likely to be critical to success. A rule-based password expansion could also add significant value to that process.

Any addresses and public keys identified by the QR code capture or keyword search process will be added to the list of found public keys and addresses. These are not as significant as keys and addresses associated with a specific wallet. Seed phrases found will be added to a list of seed phrases to be passed onto the analysis phase.

3.3. Analysis

The analysis phase will be predominately conducted on a cloud-based computing platform but will be orchestrated from the investigator's laptop. This allows for scalability and massive parallelisation. The amount of computing power that can be brought to solve the problem is effectively limited by cost only. It is trivial to split up a dictionary across x platforms to reduce the processing time. Total time $= \frac{(p \times t)}{x}$, where t is the time taken per permutation and p is the total number of permutations. Fig. 12 shows an overview of the analysis phase.

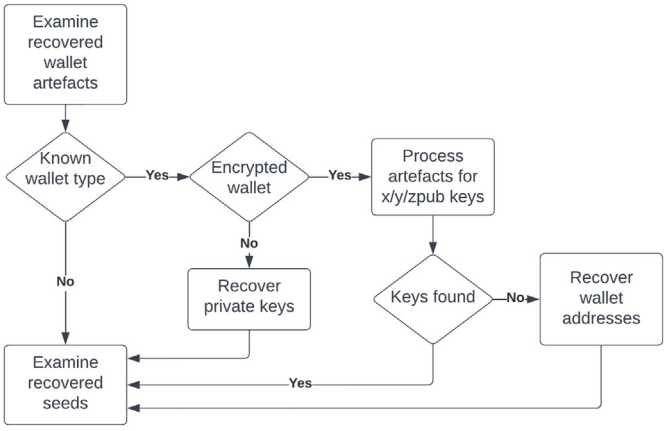

Fig. 10. Examination phase one.

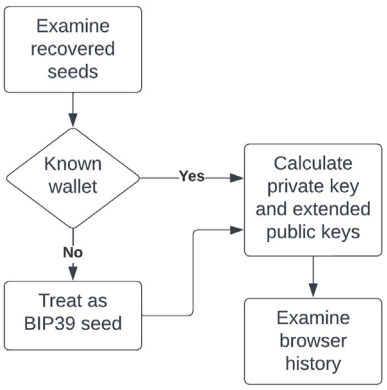

Fig. 11. Examination phase two.

First, analysis of the browser history and browser credentials for hosted wallets and associated usernames and passwords is conducted, which may give the investigator a quick win by identifying a hosted wallet.

The next part of the analysis is to establish if any of the recovered wallets are worth attacking before an attempt is made. This may not be possible and depends on the artefacts that were recovered by the collection and examination phase. This is done by identifying private and extended public keys that are associated with a wallet application and using these to generate the wallets addresses. The balance can then be obtained through a Blockchain API call. Depending on the associated wallet, several derivation paths may need to be used to generate addresses. There are three common derivation paths, Legacy, Pay to Script Hash and SegWit. Addresses recovered that are not associated with a wallet application can also be checked for a balance. At this stage, some initial reporting needs to be made to allow the investigator to choose the next course of action and set some parameters based on the initial results.

If wallets or seeds are going to be attacked, the next step is to compile a dictionary. This will consist of a word list of common passwords, augmented with the passwords recovered from the target system in the collection phase. It could also include passwords associated with the user from published breach datasets. The password set could then be mangled to usefully increase the dictionary size further (Jourdan and Stavrou 2019). This dictionary will be used in subsequent attacks.

It is trivial to split a fixed size dictionary into x parts. With knowledge of the hashing rate, an estimate can be made for how long it will take to exhaust the dictionary given x compute platforms. Cloud compute platforms are usually billed by the hour. Therefore, the cost is similar if the process is run on one platform for 2 h or two platforms for 1 h. The size of the dictionary you can process in an hour is just a factor of how much you are willing to pay and is limited by the number of computing platforms available.

Software wallet files all have some form of checksum that allows the software to identify correctly decrypted wallet files and hence the correct password. Each wallet application uses a different encryption scheme with different password-based key derivation schemes. It is the password-based key derivation function that has the biggest impact on the speed at which passwords can be guessed

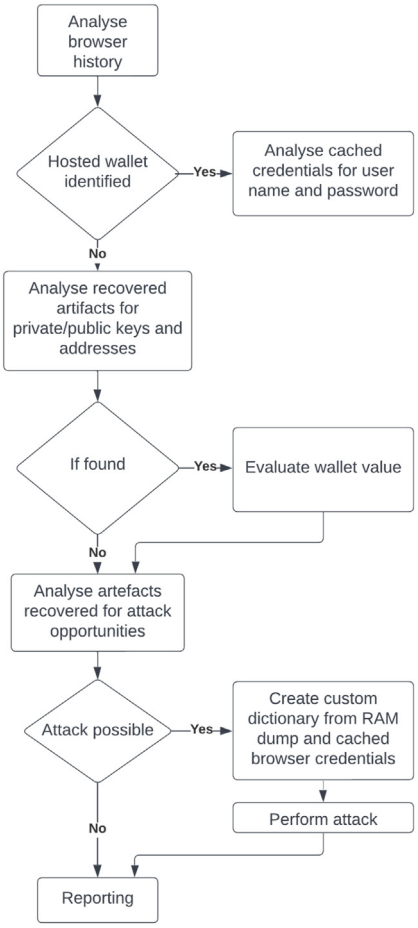

Fig. 12. The analysis phase.

in an offline password attack.

BIP39 seeds do have a checksum but this only checks the seed words for unintentional data entry errors. To identify if the correct password has been supplied you need to use the seed word and passphrase to calculate the master seed. Then from the master seed, calculate the extended public key. This can be compared to the extended public key recovered during the collection phase. If the extended public key was not found, addresses can be calculated with popular derivation paths. These addresses can then be compared with a database of all addresses on the Blockchain. The full Blockchain data is not used. Instead, an optimised database of just address hashes can be compiled ahead of time in a similar manner to Vasek et al. (2016). This reduces the size of the data and the search times.

The dictionary is split into *x* parts; each of which will be processed by a different cloud compute platform. Either the entire dictionary will be exhausted without the correct password being found, or a matching password will be found. Once the wallet password has been found, the value of the wallet can be verified using a Blockchain API call and the results presented to the investigator.

3.4. Reporting

Once the process is complete, a full report covering all four stages is presented to the investigator. This will summarise the artefacts found, the extracted identifiers, any recovered wallets along with their current value and associated keys. A full audit log will also be generated separately covering ACPO Principle Three. All the recovered files and reports will be hashed to allow the detection of unintentional changes to the files. A hash of all of the file hashes will then be created. This hash value should be made immutable by the investigator as soon as practicable, to allow detection of unintentional and malicious changes to the files.

4. Experimental methodology

The core focus of this work is it to attack a Bitcoin wallet application during a law enforcement search, to gain control of the cryptocurrency assets. It includes whether sufficient forensic artefacts be recovered to enable an attack to take place, and an attack be performed quickly enough to make it practicable during a search.

The two wallet applications that the experiments will be tested against will be Electrum (version 4.1.5) a software wallet and Ledger Nano S hardware wallet (Ledger Live version 2.34.2). The popular Electrum wallet was chosen as it is open-source and written in Python which allowed it to be trivially reverse engineered. The Ledger hardware wallet was chosen as it is the most popular hardware wallet (Thomas et al., 2020). The Ledger Live software is also open-source making it trivial to reverse engineer.

4.1. Artefact collection

The goal of the collection tool is to obtain sufficient artefacts and identifiers for an attack to be conducted. The identifiers contained in the artefacts we are looking for are Bitcoin private keys, extended public keys and addresses. We are also looking for encrypted keys. The tools constraints are: it should not unnecessarily alter data on the system, it should record a log of the actions taken and it should complete within a reasonable period.

Before the tool could be built, an examination of the forensic artefacts left by each of these two applications was conducted. This was done by creating a Windows 10 virtual machine and then taking a baseline snapshot. The applications were then installed one at a time on the baseline snapshot, during which the files and registry entries created were monitored with Microsoft's Sysinternals Process Monitor. These files and registry entries were then examined to identify reliable methods of identifying if the application was installed. In each application, a hierarchical Bitcoin wallet was created with a known seed phrase and derivation path. This allowed identifiers to be searched for, as the keys and addresses are known allowing the identification of specific file and memory structures where these are identifiers are stored. In both applications, a known password was used to encrypt the wallet data so that the plain text password could be searched for. Once this was completed, a dump of the running application process was conducted and the process dump was then examined for these identifiers. A code review of these applications was also conducted, targeting the decryption of the wallet data and, in the case of Electrum, the storage of private keys in RAM. Once the location of the relevant identifiers had been reliably found, a collection tool was designed to collect the artefacts containing these identifiers.

The collection tool was also designed to collect possible user passwords so that they can be added to any word dictionary used in the subsequent attack stages. These are extracted from cached browser passwords and the RAM dump.

The collection tool was written in PowerShell and leveraged several free or open-source utilities. Predominately open-source forensic tools are written in Python, for example, Autopsy and the Volatility Framework (Basis Technology 2021, The volatility Foundation, 2020). However, this is not the first forensic tool written in PowerShell, Barakat and Hadi (2016) developed a function-rich forensic PowerShell Module. PowerShell was chosen

as it allows for quick code revisions and did not require an interpreter to be installed on the target system. While a Python application can be compiled into a standalone executable with utilities like pyinstaller (PyInstaller Development Team, 2021), the resulting executable is often large. The completed collection tool makes use of the following utilities:

- NirSoft WebBrowserPassView
- NirSoft BrowsingHistoryView
- Microsoft Sysinternals ProcDump
- Google/Velocidex WinPmem
- Microsoft robocopy

The collection tool does not implement all of the possible functions identified in the collection framework, it does not perform a search for QR codes and does not perform a keyword search through all files in the file system. This is because both of these functions were found to be very time consuming during initial testing (Microsoft 2018). The final collection tool PowerShell script is listed in the appendix.

4.2. Artefact collection evaluation

The collection tool was tested on a freshly installed Windows 10 virtual machine. The tool is run at three different stages:

1. Before the installation of any Bitcoin wallet application.
2. After the Bitcoin wallet application has been configured and is running.
3. 15 min after the application has been terminated with the computer being left idle.

The first collection will identify false positives. The second and third collection simulates two states that the investigator may find the target system in. The wallet applications will be installed with the default options. A known wallet will then be restored in the application from a set recovery seed. In the case of the Ledger hardware wallet, after the device is restored with the recovery seed, the hardware device is synchronised with the Ledger Live application on the computer. Between each application being tested, the virtual machine will be reset to the baseline snapshot. The baseline snapshot was taken just after the Windows 10 installation was complete. All of the wallets were set up with the same recovery seed, "rude original bachelor leave round toss lend awful behave elite april super" and used the default address type, which for both Ledger and Electrum is a native SegWit wallet. After all three collections have been completed, the collected artefacts will be examined to identify if they contain the expected plain text or encrypted wallet files, private keys, extended public keys and addresses.

4.3. Bitcoin wallet attack development

The goal of the wallet attack tool is to identify the user-supplied password that has been used to encrypt the wallet file, encrypt the private key or protect the recovery seed. While it is impractical to use a dictionary or brute force attack against the encryption key, it is possible to leverage the flawed way people choose passwords to guess the user's password from which the encryption key is derived. The correct password is identified by making repeated guesses from a dictionary and checking the output from the decryption function for a condition that indicates success. This is referred to as an offline dictionary attack.

To develop the tools, the artefacts found during the collection phase were analysed to identify possible attack vectors. These included the encrypted wallet file from Electrum, the encrypted private keys recovered from the process memory and password protected recovery seeds. Some of the identifiers found within the recovered artefacts were found to assist with identifying the success condition, for example, the extended public key recovered from the Ledger Live application.

The literature review identified several tools that could conduct dictionary attacks on encrypted Electrum wallet files, including Hashcat. Where a tool existed it was evaluated for performance. Where a tool did not exist a custom tool was developed using Python. The Python module PyOpenCL was used to support writing performance-critical sections of the code in OpenCL to run on a Graphics Processing Unit (GPU). Code reviews of the open-source applications were conducted to reverse engineer the encryption scheme and file formats. Rather than writing the OpenCL cryptographic primitives from scratch, these were borrowed from Hashcat. This allowed for quick development and also allowed the tool to leverage the highly optimised Hashcat primitives. The tools were then developed using these cryptographic primitives to implement the reverse-engineered encryption scheme. A large password dictionary was also created to test the functionality and performance of the tools.

Hashcat was tested against the recovered encrypted wallet file created by the Electrum application, and two custom attack tools were developed. The first targets an encrypted private key found in the process dump of the Electrum application. The second targets a recovered BIP39 seed phrase protected by a password. This tool leverages the extended public key recovered from the Ledger Live application to identify the success condition.

4.4. Bitcoin wallet attack evaluation

To test the practicality of a dictionary attack against the wallet, we will use the artefacts and identifiers recovered by the collection tool tests with Hashcat or our custom written tools to try to recover the user password protecting the wallet file, encrypted private key or seed phrase. The experiment will run the wallet recovery tool to evaluate how many passwords can be guessed in a given period on two distinct platforms. A typical laptop an examiner may use during a search, in this case, a Dell 7390 laptop with an integrated Intel UHD 620 GPU. Secondly, a cloud platform with a high-end consumer GPU, an NVIDIA RTX3090. The tools are then assessed on these platforms to see how many passwords can be guessed per second. To calculate the password rate, the tool was timed processing a dictionary containing 64,000,000 passwords. Then the number of guesses in an hour for a fixed cost of £25 is calculated based on the tool running across a collection of hired cloud GPU instances.

5. Results

This section first sets out the results from the forensic examination of the Electrum and Ledger Nano S wallet applications. These results identify what Bitcoin artefacts and identifiers can be recovered and, how they assist with any subsequent attacks against the wallet. Second, we document the artefact collection tool evaluation, which identifies if the collection tool correctly collects the available artefacts and identifiers. Finally, we present the results of the dictionary-attack tools in terms of their password guessing rate and calculate how many guesses can be made in 1 h for £25 using a public cloud compute platform. This provides an insight into how practicable a particular attack is.

5.1. Collection tool electrum

Evaluation of the artefacts collected before the installation of the Electrum wallet application shows the recovery of no wallet files, extended public keys, extended private keys or addresses. This is the expected outcome. Any results at this stage would be false positives.

Evaluation of the artefacts collected while the Electrum wallet application was running, showed the tool recovered the encrypted wallet file from the disk. The tool also recovered an extended public key, an encrypted extended private key and addresses associated with the wallet from the Electrum process memory dump. These values are stored in a Python nested dictionary structure in the Electrum process memory. There are two substructures of interest the "keystore" shown in Fig. 13 and "addresses". The addresses structure contains a list of receiving and change addresses used by the wallet. These addresses can be quickly checked using a commercial Blockchain API for the balance and transaction history. The wallet password and the recovery seed phrase were not recovered from the Electrum process memory dump.

The Keystore structure was found to contain the extended public key and the deterministic wallet derivation path, these can be used to generate addresses associated with the wallet for Blockchain API queries. There is also an entry for the extended private key which appears to be encrypted. Examination of the Electrum source code shows that this is encrypted with AES256-CBC using the wallet password. The decryption process as pseudocode is shown in Fig. 14. The code highlights a second possible approach to conducting a dictionary attack on an Electrum wallet if the encrypted extended private key is recovered. This would cost two SHA256 operations and a single block AES decrypt. This compares to a PBKDF2-HMAC-SHA512 with 1024 iterations for the wallet file. Therefore attacking the encrypted extended private key should be orders of magnitude faster than attacking the wallet file. However, the encrypted extended private key was only obtained when the Electrum application was running.

Evaluation of the artefacts recovered 15 min after the Electrum application was terminated, identified that the encrypted wallet file was still recovered. However, no extended public keys, encrypted extended private keys or addresses associated with the wallet were found in the memory dump. The password was also not located in the memory dump. A summary of the results is shown in Table 2.

The results show that we can only obtain the extended public key, addresses and encrypted extended private key if the Electrum application is running, otherwise, all we can obtain is the encrypted wallet file. No memory artefacts appear to persist for any length of time after the application is terminated.

The collection tool was profiled with Microsoft Sysinternals Process Monitor to identify how much RAM was used during execution. The logs also show what file and registry entries were changed during its execution. A screenshot showing the memory usage for the tool is shown in Fig. 15.


```
"keystore": {"type": "bip32",
"pw_hash_version": 1,
"xpub":"zpub6rdRcr57RL1TuSXFcGyTC1nMLBB7Udsx2oqL9dXbiMG585uGsS
uT6gGzLckpGSwRvtmqpUEJAXhcQD2exFyDiUiyX1Am4eJ4wMZmNmZzSH6",
"xprv": "oChWWzfakmOz8E4tJewcUCW3k7iwKN3rZoFO2vqoF+MxRPP0VQRIPs
MeIwdG9aLWpD9kpf7O0jidsmlTUXKXLiStfmuCFFNktzBpcXnFZ7Sp9lULEOGgw
NGJo90B61jf9B63qj1dsyL+FGUetgGc7cO6JU6m7bsEVN9YC9hQNlE=",
"derivation": "m/84'/0'/0'",
"root_fingerprint": "f202e755"}
```


Fig. 13. The keystore, extracted from the process memory dump.

```
data = base64_decode(xprv)
key = sha256(sha256(password))
iv, cyphertext = data[:16], data[16:]
plaintext = AES256_CBC_Decrypt(key,iv,cyphertext)
```

Fig. 14. Decryption of the extended private key (xprv) in python.

5.2. Collection tool ledger

Evaluation of the artefacts collected before the installation of the Ledger Live application shows the recovery of no extended private keys, extended public keys or addresses. This is the expected outcome. Any results at this stage would be false positives.

The artefacts collected while the Ledger Live application was running and had been synchronised with a Ledger Nano S hardware device, include the extended public key and some addresses. These were recovered from the Ledger Live process memory dumps. However, these are also available in the file "%APPDATA%\Ledger Live\app.json". This file contains a wealth of information, including identifiers for the hardware devices being used, the extended public key, the derivation path, addresses, current balance and historic balances over the preceding 12 months. A snippet of this file is shown in Fig. 16. If the Ledger Live application is protected with an optional password the public extended key is encrypted in the app.json file. Reviewing the source code shows that it is encrypted with AES-256-CBC using PBKDF2-SHA512 with 10,000 iterations to stretch the key. This would make it relatively slow to conduct a dictionary attack against it. If the process is running and the password has been entered the extended public key can be recovered from the process memory dump. The password used to decrypt the extended public key was also found several times in the process memory dump taken while the application was running.

Evaluating the artefacts recovered 15 min after the Ledger Live application was closed, identified that the "%APPDATA%\Ledger Live\app.json" file was still recovered. However, none of the artefacts found earlier in the process memory dump could be found in the RAM dump. A summary of the results are shown in Table 3.

The results show that we can obtain the extended public key and associated addresses during and after the execution of the Ledger Live application as long as the optional password feature has not been used.

5.3. Electrum wallet file dictionary attack

Conducting a dictionary attack on an Electrum wallet file can be done with existing open-source tools. First, a hash compatible with Hashcat (2021) is created using electrum2john.py (Kholia 2021) part of the John the Ripper jumbo release (Open Wall 2021). Then Hashcat is used with a password dictionary to guess the password. On a typical contemporary laptop, for example, a Dell 7390 with 16 GB of RAM, an i7-8650U CPU and an Intel UHD 620 GPU, Hashcat can check approximately 3300 passwords per second. On a state-of-the-art consumer GPU for example, an NVIDIA GeForce RTX 3090, Hashcat can check approximately 880,000 passwords per second. Therefore the laptop could check just under thirteen million (12,880,000) passwords an hour. The RTX 3090 could check just over three billion (3,168,000,000) passwords per hour. Table 4 shows the time it would take to exhaust popular dictionaries using this attack.

It was identified that the collection tool also recovered an encrypted extended private key from the Electrum process memory. This can be attacked as well. It can be seen from Fig. 14 that this is protected by two SHA256 operations and a single AES-256 block decrypt operation. This should be orders of magnitude faster to process than the wallet file, protected by PBKDF2-HMAC-SHA512

Table 2
Collection tool results for electrum.

Information Recovered	Before Installation	During Execution	After Execution
wallet file (encrypted)	X	✓	✓
private key (encrypted)	X	✓	X
private key (plaintext)	X	X	X
public key	X	✓	X
wallet password	X	X	X
recovery seed	X	X	X
addresses	X	✓	X

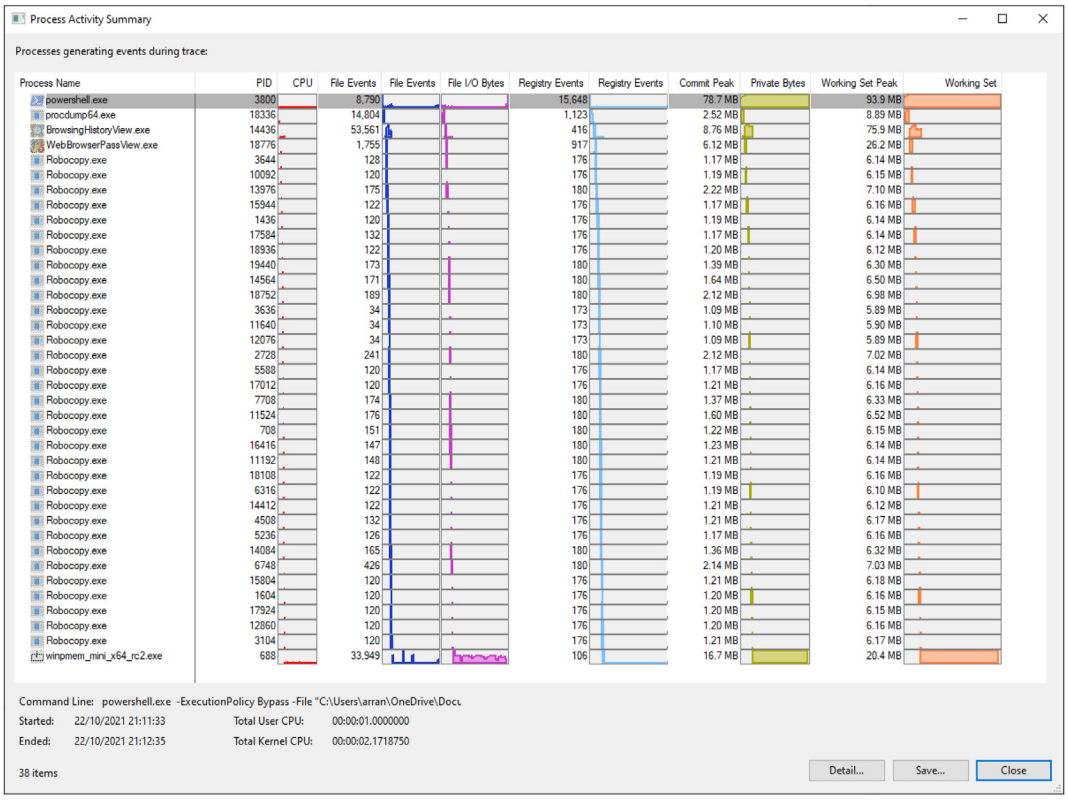

Fig. 15. Collection tool profiling.

```
"accounts": [
    {
      "data": {
        "id": "libcore:1:bitcoin:xpub6Cxu1WjH7xvWCr91wZQCmqbLzEtDbPtxCant
        aqjpxLWK1tGpN8aKrYxiJCqeGddb7cYEKX3BFCzWddoXWs9C81MmnKmutpf6PuSU
        baLCWop:native_segwit",
        "seedIdentifier": "048a0d3ce3dd74cab35f532d63218dab1a6f54395764b7
        f24d12a38f94e1b72a30b43c2f2f5ebdbd358ef6f332a9fd2ec92d86a55f01dac
        1c17b8b9fa69688765c",
        "name": "Bitcoin 1",
        "starred": false,
        "used": false,
        "derivationMode": "native_segwit",
        "index": 0,
        "freshAddress": "bc1qv7wymgqr2qr64czhdyl5cgykrsgzjtmd5h2zn6",
        "freshAddressPath": "84'/0'/0'/0/0",}}
```

Fig. 16. A Snippet from app.json.

with 1024 iterations. Initially, a C program was written using the OpenSSL library for the AES and SHA256 cryptographic primitives. This was tested and performed 1.7 million passwords guesses per second using a single CPU thread on the laptop described above. Next, a Python version was developed using pyOpenCL to leverage the GPU. This used the OpenCL cryptographic primitives library from Hashcat (2021). The biggest bottleneck found was Python loading passwords from a password file and formatting them in memory to present to the GPU. This was solved by formatting the dictionary in a pre-process so that it could be read straight into memory and passed to the GPU in blocks of 10 million passwords. This pre-processing traded an increased dictionary size in bytes for speed. The tool was tested on the same laptop and performed 3.9 million guesses per second using the integrated GPU. The code design takes a few shortcuts to increase guesses per second. First, the AES decryption is only performed on the first 16 bytes (1 AES Block) of the encrypted extended private key. To identify success, the first 4 bytes of the decrypted result is checked for the string xprv, yprv or zprv. While this is faster it does generate some false positives. However, after candidate passwords have been identified they can be verified by performing the full decryption and verifying the Base58Check checksum. Running on an Nvidia RTX 3090 the tool performed about 16.8 million password guesses per second. This is almost 20 times faster than attacking the wallet file on the

Table 3
Collection tool results for ledger live.

Information Recovered	Before Installation	During Execution	After Execution
wallet file (encrypted)	N/A	N/A	N/A
private key (encrypted)	X	X	X
private key (plaintext)	X	X	X
public key	X	✓	✓
wallet password/PIN	X	✓	X
recovery seed	X	X	X
addresses	X	✓	✓

Table 4
Dictionary and the Time to Exhaust them for Electrum wallets.

Dictionary	Number of passwords	Wallet attack time (RTX3090)
Rockyou	14,344,392	16.3 s
Crackstation	63,941,070	1.21 min
All-in-One-P	15,462,473,182	4.88 h
a-zA-Z0-9 {8}	96,717,311,574,016	3.49 years

same hardware. Table 5 shows the time it would take to exhaust popular dictionaries using this attack.

5.4. Ledger recovery seed dictionary attack

The examination of the artefacts created by the Ledger Live application did not identify the recovery seed at any point. Ledger uses a standard BIP39 recovery seed. The extended public key assists with an attack as it can be used to identify success. It is not practicable to guess the recovery seed due to the size of the search space which is approximately 2048^{11} possibilities. However, if the recovery seed is found, for example in a physical search, but it has been protected with a password and the extended public key can be recovered from the target computer, it may be practicable to conduct a dictionary attack to recover the private key.

BTCrecover has functionality for conducting dictionary attacks against BIP39 recovery seeds and it supports GPU acceleration. However, this functionality is reported as being experimental by the developer. When it was tested it failed to run on the test system. Instead, a Python application using the OpenCL and cryptographic primitives from Hashcat was developed. This application used the extended public key recovered by the collection tool from the Ledger Live application to identify success. The application was run on the test laptop, which managed 1200 passwords per second. On the NVIDIA RTX 3090, it performed 115,500 passwords per second. Table 6 shows the time it would take to exhaust popular dictionaries using this attack.

The previous results are all for a single GPU instance. However, the problem is trivial to break up into parallel processes that could run on separate GPU instances. The dictionary can be split into chunks with each GPU instance processing a different chunk. As we saw in Fig. 7 the cost of a GPU instance on vast.ai is quite low at around $0.60 per hour. So for about £25 you could hire 40 RTX 3090 GPU instances for an hour. Table 7 shows the approximate number of passwords that could be guessed for £25.

Table 5
Dictionary and the Time to Exhaust them for Electrum Key Attack.

Word list	Number of passwords	Wallet attack, single RTX3090
Rockyou	14,344,392	0.85 s
Crackstation	63,941,070	3.8 s
All-in-One-P	15,462,473,182	15.31 min
a-zA-Z0-9 {8}	96,717,311,574,016	66.5 days

The results show that it is likely to be practicable to conduct a dictionary attack against Electrum and Ledger Live under specific circumstances. Where an encrypted extended public key can be recovered from a running Electrum process 2.4 trillion guesses can be made in an hour costing just £25 to hire the necessary GPU instances. Where only the wallet file was recovered we were still able to make 126 billion guesses in an hour again for approximately £25. Where a Ledger recovery seed has been found and the extended public key recovered 16.6 billion guesses can be made in an hour for approximately £25.

6. Conclusions

This paper proposed a framework for live host-based bitcoin wallet forensic triage and evaluated the practicability of using a dictionary attack against an Electrum and Ledger Bitcoin wallet during a police search, with the intention of seizing the Bitcoin under Proceeds of Crime Act (POCA) legislation. The ethics of cracking a Bitcoin wallet should, of course, be strongly considered, and could probably only be justified within serious criminal activity.

Based on the literature review, a triage framework for the recovery of Bitcoin artefacts during a search was developed. To answer the research questions, tool were developed and tested covering the collection and analysis phase of the framework. These tools were evaluated against the two different wallet solutions, the Electrum software and Ledger Nano S hardware wallet.

To develop the collection tool first, the research work in this paper conducted a forensic investigation of the artefacts created by the two popular Bitcoin wallet solutions. From this work, artefacts were identified that could either help establish the balance of the wallet or could assist in a dictionary attack against the wallet. Using the results of this investigation a PowerShell script was developed to recover useful artefacts from a target computer, taking advantage of several 3rd party utilities. This tool was evaluated to answer the first research question.

Finally, three proof of concept dictionary attack tools were developed to attack both the Electrum and Ledger Nano S wallets. These tools use the artefacts collected by the collection tool.

Declaration of competing interest

The authors whose names are listed in the paper certify that they have NO affiliations with or involvement in any organization

Table 6
Dictionaries and the Time to Exhaust them for Ledger.

Dictionary	Number of passwords	Wallet attack on RTX3090
Rockyou	14,344,392	2.07 min
Crackstation	63,941,070	9.22 min
All-in-One-P	15,462,473,182	37.19 h
a-zA-Z0-9 {8}	96,717,311,574,016	26.55 years

Table 7
The number of guesses achievable for £25.

Target	Password Rate (Single GPU)	Total Guesses for £25
Ledger seed password	115,500	16,632,000,000
Electrum Wallet file	880,000	126,720,000,000
Electrum encrypted key	16,800,000	2,419,200,000,000

or entity with any financial interest (such as honoraria; educational grants; participation in speakers' bureaus; membership, employment, consultancies, stock ownership, or other equity interest; and expert testimony or patent-licensing arrangements), or nonfinancial interest (such as personal or professional relationships, affiliations, knowledge or beliefs) in the subject matter or materials discussed in this manuscript.

Data availability

Data will be made available on request.

Appendix

Collection Tool Source Code

Listing 1: https://github.com/billbuchanan/bitcoinframework/blob/main/runmeasadmin.bat
Listing 2: https://github.com/billbuchanan/bitcoinframework/blob/main/collect.ps1

Electrum Private Key Source Code

Listing 3: https://github.com/billbuchanan/bitcoinframework/blob/main/testElectrum.py
Listing 4: https://github.com/billbuchanan/bitcoinframework/blob/main/xprv.cl

Ledger Recovery Seed Source Code

Listing 5: https://github.com/billbuchanan/bitcoinframework/blob/main/testLedger.py
Listing 6: https://github.com/billbuchanan/bitcoinframework/blob/main/ledger_seed.cl

Hashcat Benchmarks

Listing 7: https://github.com/billbuchanan/bitcoinframework/blob/main/HashcatBenchmarks.txt

Ethics

The tools created during this paper will not be tested against actual recovered wallet files or artificially created *real world* wallets. This is because of the risk of using recovered wallets presents to the assets contained in the wallet. The second reason is that, if the wallet is created specifically for the experiment by a volunteer, there are two possible undesirable outcomes. The volunteer generates realistic passwords possibly giving away information about how they construct their real-world passwords or they create an artificial password that is unrealistic.

While writing this paper, a weakness was discovered in the way Electrum stores the extended private key. The key is stored in memory encrypted with the user's password. However, the key derivation can be performed very quickly. This allows an attacker to speed up password attacks in specific circumstances. This is not a bug; it is a deliberate design decision made by the developers. Therefore, it has not been reported as a vulnerability.

References

Akhgar, B., Wells, D., 2018. 'Critical success factors for osint driven situational awareness'. European Law Enforcement Research Bulletin 18. Retrieved August 1, 2021 from. https://bulletin.cepol.europa.eu/index.php/bulletin/article/view/332/289.

Barakat, A., Hadi, A., 2016. Windows forensic investigations using powerforensics tool. In: '2016 Cybersecurity and Cyberforensics Conference (CCC)'. IEEE, pp. 41–47.

Basis Technology, 2021. Autopsy digital forensics. Retrieved August 26, 2021 from. https://www.autopsy.com/.

Bayne, E., Ferguson, R.I., Sampson, A., 2018. Openforensics: a digital forensics gpu pattern matching approach for the 21st century. Digit. Invest. 24, S29–S37.

Bell, G.B., Boddington, R., 2010. Solid state drives: the beginning of the end for current practice in digital forensic recovery? J. Digit. Foren. Secur. Law 5 (3), 1.

blockchain.com, 2021. Blockchain size (MB). Retrieved October 10, 2021 from. https://www.blockchain.com/charts/blocks-size.

Brown, D., 2010. Sec 2: Recommended elliptic curve domain parameters. Retrieved July 17, 2021 from. https://www.secg.org/sec2-v2.pdf.

Brown, J., 2011. Zbar bar code reader. Retrieved August 15, 2021 from. http://zbar.sourceforge.net/.

Carrier, B.D., 2006. Risks of live digital forensic analysis. Commun. ACM 49 (2), 56–61.

Case, A., Richard III, G.G., 2017. Memory forensics: the path forward. Digit. Invest. 20, 23–33.

Chainalysis, 2018. 'Mapping the universe of bitcoin's 460 million addresses'. Retrieved August 7, 2021 from. https://blog.chainalysis.com/reports/bitcoin-addresses.

Chistyakova, Y., Wall, D.S., Bonino, S., 2019. 'The back-door governance of crime: confiscating criminal assets in the UK'. Eur. J. Crim. Pol. Res. 1–21.

Chivers, H., Hargreaves, C., 2011. Forensic data recovery from the windows search database. Digit. Invest. 7 (3–4), 114–126.

Coinbase (n.d.), 'Does coinbase freeze accounts?'. Retrieved August 7, 2021 from https://help.coinbase.com/en/coinbase/other-topics/other/does-coinbase-freeze-accounts.

Courtois, N., Song, G., Castellucci, R., 2016. Speed optimizations in bitcoin key recovery attacks. Tatra Mount. Mathem. Publ. J. Slovak Acad. Sci. 67 (1), 55–68.

Essex Police, 2018. POLICY – proceeds of crime, number:S1150. Retrieved July 21, 2021 from. https://www.essex.police.uk/SysSiteAssets/foi-media/essex/our-policies-and-procedures/serious-crime-directorate/s1150-policy-proceeds-of-crime.pdf.

Foley, S., Karlsen, J.R., Putniņš, T.J., 2019. Sex, drugs, and bitcoin: how much illegal activity is financed through cryptocurrencies? Rev. Financ. Stud. 32 (5), 1798–1853.

Forte, D., 2004. The importance of text searches in digital forensics. Netw. Secur. 2004 (4), 13–15.

Gabbatt, A., 2021. How the colonial pipeline hack is part of a growing ransomware trend in the us. Retrieved July 21, 2021 from. https://www.theguardian.com/technology/2021/may/19/colonial-pipeline-cyber-attack-ransom.

Hampton, N., Baig, Z.A., 2015. Ransomware: emergence of the cyber-extortion menace. In: AISM 2015 : Proceedings of the 13th Australian Information Security Management Conference, pp. 47–56.

Hashcat, 2021. Hashcat - advanced password recovery. Retrieved August 7, 2021 from. https://hashcat.net/hashcat/.

Heutmaker, K., 2017. Multibit is deprecated - do not use. Retrieved August 7, 2021 from. https://github.com/Multibit-Legacy/multibit.

Holm, H.H., Brodtkorb, A.R., Sætra, M.L., 2020. Gpu computing with python: performance, energy efficiency and usability. Computation 8 (1), 4.

Home Office, 2018. Code of practice issued under section 47s of the proceeds of crime act 2002. Retrieved August 1, 2021 from. https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment_data/file/678000/CCS207_CCS0118810738-1_HO_POCA_COP_Search_Seizure_Detention_Accessible.pdf.

Home Office, 2019. Asset recovery action plan. Retrieved July 21, 2021 from. https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment_data/file/815900/20190709_Asset_Recovery_Action_Plan_FINAL_Clean.pdf.

Jafari, F., Satti, R.S., 2015. Comparative analysis of digital forensic models. J. Adv. Comput. Netw. 3 (1), 82–86.

Jourdan, P., Stavrou, E., 2019. Towards designing advanced password cracking toolkits: optimizing the password cracking process. In: Adjunct Publication of the 27th Conference on User Modeling, Adaptation and Personalization, pp. 203–208.

Kent, K., Chevalier, S., Grance, T., Dang, H., 2006. Guide to integrating forensic techniques into incident response. NIST Spec. Publ. 10 (14), 800–886.

Kent Police (n.d.), 'Investigate first - progression opportunities'. Retrieved July 21, 2021 from https://www.kent.police.uk/police-forces/kent-police/areas/kent-police/c/careers/police-officers/new-investigate-first/progression-opportunities/.
Kholia, D., 2021. electrum2john.py. Retrieved August 26, 2021 from. https://github.com/openwall/john/blob/bleeding-jumbo/run/electrum2john.py.
Ledger (n.d.), 'Hardware wallet comparison'. Retrieved August 7, 2021 from https://shop.ledger.com/pages/hardware-wallets-comparison.
Levi, M., 1997. Taking the profit out of crime: the UK experience. Eur. J. Crime Crim. Law Crim. Justice 5, 228.
Microsoft, 2018. SetFileTime function (fileapi.h). Retrieved September 18, 2021 from. https://docs.microsoft.com/en-gb/windows/win32/api/fileapi/nf-fileapi-setfiletime?redirectedfrom=MSDN.
Nakamoto, S., 2008. Bitcoin: a peer-to-peer electronic cash system. Decentral. Bus. Rev., 21260
National Crime Agency, 2019. Defence against money laundering (DAML). Retrieved October 6, 2021 from. https://www.nationalcrimeagency.gov.uk/who-we-are/publications/167-defence-against-money-laundering-daml-faq-may-2018/file.
National Crime Agency, 2021. Suspicious activity reports. Retrieved October 6, 2021 from. https://www.nationalcrimeagency.gov.uk/what-we-do/crime-threats/money-laundering-and-illicit-finance/suspicious-activity-reports.
Open Wall, 2021. John the Ripper password cracker. Retrieved August 26, 2021 from. https://www.openwall.com/john/.
Palatinus, M., Rusnak, P., Voisine, A., Bowe, S., 2013. Mnemonic code for generating deterministic keys. Retrieved August 1, 2021 from. https://github.com/bitcoin/bips/blob/master/bip-0039.mediawiki.
Parlika, R., Pratama, A., 2021. Use of the web api as a basis for obtaining the latest data on bitcoin prices at 30 exchange places. In: IOP Conference Series: Materials Science and Engineering, vol. 1125. IOP Publishing, 012035.
Pearman, S., Thomas, J., Naeini, P.E., Habib, H., Bauer, L., Christin, N., Cranor, L.F., Egelman, S., Forget, A., 2017. Let's go in for a closer look: observing passwords in their natural habitat. In: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, pp. 295–310.
Povey, K., Wooler, S., Dixon, S., 2004. Joint review of asset recovery since the proceeds of crime act 2002. Retrieved July 21, 2021 from. https://www.justiceinspectorates.gov.uk/cjji/inspections/payback-time-joint-review-of-asset-recovery-since-the-proceeds-of-crime-act-2002/.
Proceeds of Crime Act, 2002. Retrieved July 17, 2021 from. https://www.legislation.gov.uk/ukpga/2002/29/contents.
PyInstaller Development Team, 2021. PyInstaller. Retrieved September 17, 2021 from. http://www.pyinstaller.org/.
Rafique, M., Khan, M., 2013. Exploring static and live digital forensics: methods, practices and tools. Int. J. Sci. Eng. Res. 4 (10), 1048–1056.
Richardson, R., North, M.M., 2017. Ransomware: evolution, mitigation and prevention. Int. Manag. Rev. 13 (1), 10.
Rothery, S., 2021. BTCrecover. Retrieved August 7, 2021 from. https://btcrecover.readthedocs.io/en/latest/.
Ruddick, A., Yan, J., 2016. Acceleration attacks on pbkdf2: or, what is inside the black-box of oclhashcat?. In: 10th {USENIX} Workshop on Offensive Technologies ({WOOT} 16).
Rusnak, P., 2013. mnemonic.py. Retrieved August 1, 2021 from. https://github.com/trezor/python-mnemonic/blob/master/src/mnemonic/mnemonic.py.
Russinovich, M., Richards, A., 2021. ProcDump v10.1. Retrieved August 15, 2021 from. https://docs.microsoft.com/en-us/sysinternals/downloads/procdump.
San Pedro, M., Servant, V., Guillemet, C., 2019. Practical side-channel attack on a security device. In: 2019 31st International Conference on Microelectronics (ICM). IEEE, pp. 130–133.
Scan Computers International Ltd, 2021. GeForce RTX 3090 graphics cards. Retrieved September 15, 2021 from. https://www.scan.co.uk/shop/computer-hardware/gpu-nvidia-gaming/nvidia-geforce-rtx-3090-graphics-cards.
Shannon, C.E., 1948. A mathematical theory of communication. Bell Syst. Tech. J. 27 (3), 379–423.
Sofer, N. (n.d.), 'Nirsoft Utilities'. Retrieved August 15, 2021 from https://www.nirsoft.net/.
The volatility Foundation, 2020. The volatility foundation - open source memory forensics. Retrieved August 15, 2021 from. https://www.volatilityfoundation.org/.
Thomas, T., Piscitelli, M., Shavrov, I., Baggili, I., 2020. Memory foreshadow: memory forensics of hardware cryptocurrency wallets—a tool and visualization framework. Forensic Sci. Int.: Digit. Invest. 33, 301002.
Tziakouris, G., 2018. Cryptocurrencies—a forensic challenge or opportunity for law enforcement? an interpol perspective. IEEE Secur. Priv. 16 (4), 92–94.
Van Der Horst, L., Choo, K.-K.R., Le-Khac, N.-A., 2017. Process memory investigation of the bitcoin clients electrum and bitcoin core. IEEE Access 5, 22385–22398.
Vasek, M., Bonneau, J., Castellucci, R., Keith, C., Moore, T., 2016. The bitcoin brain drain: examining the use and abuse of bitcoin brain wallets. In: International Conference on Financial Cryptography and Data Security. Springer, pp. 609–618.
Visconti, A., Bossi, S., Ragab, H., Calò, A., 2015. On the weaknesses of pbkdf2. In: International Conference on Cryptology and Network Security. Springer, pp. 119–126.
Visconti, A., Mosnáček, O., Brož, M., Matyáš, V., 2019. Examining pbkdf2 security margin—case study of luks. J. Inf. Secur. Appl. 46, 296–306.
Volety, T., Saini, S., McGhin, T., Liu, C.Z., Choo, K.-K.R., 2019. Cracking bitcoin wallets: I want what you have in the wallets. Future Generat. Comput. Syst. 91, 136–143.
Wang, C., Jan, S.T., Hu, H., Bossart, D., Wang, G., 2018. The next domino to fall: empirical analysis of user passwords across online services. In: Proceedings of the Eighth ACM Conference on Data and Application Security and Privacy, pp. 196–203.
Weber, J., Kruisbergen, E.W., 2019. Criminal markets: the dark web, money laundering and counterstrategies-an overview of the 10th research conference on organized crime. Trends Organ. Crime 22 (3), 346–356.
Weir, M., Aggarwal, S., De Medeiros, B., Glodek, B., 2009. Password cracking using probabilistic context-free grammars. In: 2009 30th IEEE Symposium on Security and Privacy. IEEE, pp. 391–405.
Williams, J., 2012. ACPO good practice guide for digital evidence. Retrieved August 1, 2021 from. https://library.college.police.uk/docs/acpo/digital-evidence-2012.pdf.
Wuille, P., 2012. Hierarchical deterministic wallets. Retrieved August 1, 2021 from. https://github.com/bitcoin/bips/blob/master/bip-0032.mediawiki.
Zhang, Y., Yang, B., Rogers, M., Hansen, R.A., 2015. Forensically sound retrieval and recovery of images from gpu memory. In: International Conference on Digital Forensics and Cyber Crime. Springer, pp. 53–66.
Zollner, S., Choo, K.-K.R., Le-Khac, N.-A., 2019. An automated live forensic and postmortem analysis tool for bitcoin on windows systems. IEEE Access 7, 158250–158263.

Glossary

ACPO: Association of Chief Police Officers. 3, 4, 10, 15
API: Application Programming Interface. 4, 9, 10, 13–15, 18
ASIC: Application Specific Integrated Circuit. 9
BIP: Bitcoin Improvement Proposal. 6, 8–10, 13, 15, 17, 20
CPU: Central Processing Unit. 8, 9, 19, 20
CUDA: Compute Unified Device Architecture. 9
DAML: Defence Against Money Laundering. 1
GPU: Graphics ProcessingUnit. 1, 3, 8, 9, 12, 13, 17, 19-21
GREP: Global Regular Expression Print. 13
NCA: National Crime Agency. 1
OpenCL: Open Computing Language. 9
P2PKH: Pay to Public-key Hash. 5
PBKDF2: Password-Based Key Derivation Function Two. 6, 8, 9
POCA: Proceeds of Crime Act. 1, 21
RAM: Random Access Memory. 3, 4, 6–8, 11, 13, 16, 18, 19
SAR: Suspicious Activity Report. 1