electronics

MDPI

Article

A Security Analysis of Cryptocurrency Wallets against Password Brute-Force Attacks

Hyeonsu Byun [1], Jueun Kim [1], Yunseok Jeong [1], Byoungjin Seok [2], Seonghyeon Gong [3] and Changhoon Lee [1,*]

[1] Department of Computer Science and Engineering, Seoul National University of Science and Technology, Seoul 01811, Republic of Korea; bhsu828@seoultech.ac.kr (H.B.); jueunk0104@gmail.com (J.K.); tjr6613@seoultech.ac.kr (Y.J.)
[2] School of Cybersecurity, Korea University, Seoul 01811, Republic of Korea; bjseok@korea.ac.kr
[3] Department of Electrical & Computer Engineering, Illinois Institute of Technology, Chicago, IL 60189, USA; sgong3@iit.edu
* Correspondence: chlee@seoultech.ac.kr

Abstract: Currently, the monetary value of cryptocurrencies is extremely high, leading to frequent theft attempts. Cyberattacks targeting cryptocurrency wallets and the scale of these attacks are also increasing annually. However, many studies focus on large-scale exchanges, leading to a lack of research on cryptocurrency wallet security. Nevertheless, the threat to individual wallets is real and can lead to severe consequences for individuals. In this paper, we analyze the security of the open-source cryptocurrency wallets Sparrow, Etherwall, and Bither against brute-force attacks, a fundamental threat in password-based systems. As cryptocurrency wallets use passwords to manage users' private keys, we analyzed the private key management mechanism and implemented a password verification oracle. We used this oracle for brute-force attacks. We identified the private key management mechanism by conducting a code-level investigation and evaluated the three wallets' security through practical experimentation. The experiment results revealed that the wallets' security, which depends on passwords, could be diminished due to the password input space and the configuration of password length settings. We propose a general methodology for analyzing the security of desktop cryptocurrency wallets against brute-force attacks and provide practical guidelines for designing secure wallets. By using the analysis methods suggested in this paper, one can evaluate the security of wallets.

Keywords: cryptocurrency; cryptocurrency wallet; crypto wallet; Sparrow; Etherwall; Bither; security analysis; brute-force; password

Citation: Byun, H.; Kim, J.; Jeong, Y.; Seok, B.; Gong, S.; Lee, C. A Security Analysis of Cryptocurrency Wallets against Password Brute-Force Attacks. *Electronics* **2024**, *13*, 2433. https://doi.org/10.3390/electronics13132433

Academic Editor: Andreas Mauthe

Received: 30 May 2024
Revised: 17 June 2024
Accepted: 19 June 2024
Published: 21 June 2024

1. Introduction

Cryptocurrencies are digital assets that operate on cryptographically secure distributed ledgers. The global cryptocurrency market cap exceeds USD 829.3 billion [1]. The cryptocurrency market has consistently grown since Bitcoin emerged [2]. On 14 November 2023, Bitcoin's (BTC) market capitalization exceeded USD 71 billion [3]. Cryptocurrencies are no longer confined to the virtual realm and are actively utilized in actual asset transactions. Many investors collect cryptocurrencies, and countries like El Salvador now recognize them as legal tender [4].

Cryptocurrency operates based on public key algorithms [5], using private keys to prove ownership or conduct transactions. However, the length and randomness of private keys make them hard to remember, leading to the emergence of cryptocurrency wallets to facilitate the convenient use of digital assets. Cryptocurrency wallets protect and manage sensitive data using a user-defined password, including mnemonic codes and the user's private key. Utilizing a cryptocurrency wallet enables convenient management of digital assets and facilitates transactions.

As the value of digital assets has increased, the threats to digital assets have also grown [6,7]. According to Chainalysis, cryptocurrency thefts resulted in approximately USD 3.8 billion in losses in 2022, significantly increasing thefts and hacking incidents [8]. Atomic Wallet, with over five million users, fell victim to a hack in June 2023. Many users are suffering forced withdrawals of virtual assets from their accounts. Estimated losses exceed USD 35 million [9]. In July 2023, a cryptocurrency theft occurred in the ETH, TRX, and BTC hot wallets of the cryptocurrency payment platform Alphapo, resulting in losses of at least USD 31 million. This incident is suspected to have been caused by a private key leak [10].

The threat to cryptocurrency wallets is increasing. However, since cryptocurrency is not legally recognized in most countries, cryptocurrency management applications are not subject to the stringent regulations that apply to traditional financial applications. This makes cryptocurrency wallets more vulnerable compared to traditional financial applications [11]. Given that virtual assets are increasingly regarded as part of tangible assets, cryptocurrency theft threatens individuals' financial stability and undermines trust in the digital economy.

Desktop cryptocurrency wallets are software applications downloaded and installed on a user's computer. These wallets store sensitive information such as private keys, transaction history, and account information as local files on the device [12]. Since wallet-related local files can be accessed at any time within the device, there is a risk of physical theft if the device is compromised by malware. Attackers can access the desktop to obtain local databases, logs, and key-related files. Insecure data storage is one of the common threats to digital wallets [13]. Additionally, even if the data are encrypted, brute-force attacks can occur because attackers can still access wallet-related files [14].

Passwords can be vulnerable to guessing attacks for various reasons. Typically, when cryptographic keys and related materials are derived from or protected by passwords, they become susceptible to these attacks [15]. For example, encryption tools like BitLocker derive encryption keys from user passwords [16]. If data protected by these keys are stolen, the strength of the password becomes critical to security. The same applies to desktop cryptocurrency wallets. Although they use robust cryptographic algorithms, the private keys are protected by encryption keys derived from passwords. Thus, weaknesses can arise from password policies and the storage methods of sensitive data [11,13].

In this paper, we conducted a security analysis of desktop cryptocurrency wallets against password brute-force attacks based on local data. We scrutinized the private key management mechanism to construct a password verification oracle. Leveraging this, we conducted practical brute-force attacks on three wallets: Sparrow, Etherwall, and Bither. We then evaluated their security. Furthermore, since password brute-force attacks are fundamentally possible on desktop cryptocurrency wallets, we propose a generally applicable methodology for analyzing resistance to brute-force attacks based on our findings. This universal analysis methodology is expected to help improve the overall security of cryptocurrency wallets, a crucial area of concern in the digital age.

Our contributions are as follows:

- We performed a security analysis of three cryptocurrency wallets—Sparrow, Etherwall, and Bither—for the first time.
- We propose a general approach for the cryptocurrency wallet security analysis of brute-force attacks on passwords.
- Through experiments, we showed that the implementation design flaws can compromise security and demonstrated the gap between theoretical and practical security.

In Section 2, we review the security analysis research on cryptocurrency wallets through relevant studies. In Section 3, we establish a methodology for analyzing the security against brute-force attacks on passwords. Section 4 analyzes the wallet's implementation flaws from a security perspective. Section 5 conducts brute-force attack experiments in a practical perspective according to the methodology described in Section 3. We then conclude the paper in Section 6.

2. Related Work

Research on the security of local data in cryptocurrency wallets can be categorized into studies focusing on mobile and desktop environments, as well as memory and disk storage. On mobile wallets, Trevor Haigh et al. [17] analyzed seven Android cryptocurrency wallets and identified some vulnerabilities. Binance, Xapo, and Coinbase were excluded as they do not store wallet-related data on physical devices. However, four of the target wallets allowed the acquisition of sensitive data. Mycelium encrypts all wallet-related data, but decryption was possible. In BitcoinWallet, all related data, including private keys, were accessible in plaintext. Bitpay and Coinpayments stored wallet keys in plaintext. Notably, Coinpayments also stored user passwords in plaintext, demonstrating significant vulnerability.

Ashish Rajendra Sai et al. [11] evaluated the security of Android wallets based on the OWASP top 10 risk factors. They divided 48 apps into three groups according to the number of downloads and conducted an initial analysis using automated tools like DroidSafe. Subsequently, they manually analyzed the four most downloaded cryptocurrency applications and four traditional banking applications in detail. The analysis revealed that cryptocurrency applications had a higher frequency of security vulnerabilities compared to banking applications. The primary issues identified were inadequate encryption and insecure data-storage methods.

Md Shahab Uddin [18] developed a semi-automated security evaluation framework for analyzing Android cryptocurrency wallets and used it to analyze 311 cryptocurrency wallet apps. Additionally, he manually examined the top 18 apps from the Google Play Store. The results revealed that 111 cryptocurrency wallets stored key-related information in plaintext. Among the top 18 apps, three stored key-revealing information in plaintext, and four encrypted key-related information using easily decipherable keys. Furthermore, many apps were found to be vulnerable to dictionary attacks and exhibited both cryptographic and generic vulnerabilities.

On desktop wallets, Wiebe Koerhuis et al. [19] investigated the data left in volatile memory, network traffic, and hard disks by the privacy-focused cryptocurrencies Monero and Verge. Using volatility to analyze memory, they obtained the wallet passphrase and mnemonic seed from Monero and the wallet passphrase from Verge. Additionally, analysis with the Bulk Extractor revealed that, when using Monero, it was possible to extract wallet addresses, transaction IDs, and transaction amounts from the disk. When using Verge, they could extract wallet addresses and public keys from the disk.

Tejaswi Volety et al. [20] conducted mnemonic seed cracking on the Multibit HD and Electrum cryptocurrency wallets. They used memory scanning to extract valid mnemonic candidates and created a dictionary. Then, they performed offline brute-force attacks to recover 12-seed combination mnemonic codes. The experiments successfully recovered mnemonic codes in some cases.

Purthani Praitheeshan et al. [21] analyzed the security of the keystore file, commonly used in Ethereum wallets, and demonstrated its vulnerabilities through experiments. If an attacker obtains the keystore file and password, he/she can access the associated Ethereum account fully. Thus, the keystore file becomes the primary target of all attacks on Ethereum. Ref. [21] used Hashcat to perform brute-force and dictionary attacks on the keystore file, achieving partial password recovery. Additionally, they found a high success rate when using brute-force attacks with masking techniques or passwords from a dictionary.

Stephan Zollner et al. [22] conducted live and postmortem forensic analyses on Bitcoin-supporting wallets, including Armory, Multibit HD, Electrum, mSIGNA, Bitpay, Copay, Bither, Bitcoin Core, and Bitcoin Knots. They successfully extracted wallet login information and mnemonic code words through real-time memory analysis. They also effectively located Bitcoin-related files on the system using file signatures and keyword searches.

The analysis results for mobile and desktop platforms, disk files, and memory are summarized in Table 1. "Data Source" refers to the location from which the data used in the analysis was obtained. This can be inferred from local disk and memory. "Data Acquisition

Windows" indicates the periods when wallet-related data can be obtained. "Always" means that data can be obtained regardless of the program's runtime. In contrast, "Temporary" means that data can only be acquired while the program is running or immediately after it has been executed. "Private Key Acquisition" denotes whether the private key or mnemonic code was obtained as a result of the analysis. "Attack Execution" refers to whether the study performed simple analysis or conducted attack experiments to obtain the private key.

As previously noted, many studies attempt to extract sensitive data, such as mnemonic codes, from memory in cryptocurrency wallets. However, due to the high volatility of memory data, the window for obtaining wallet-related information is very limited. Additionally, for a comprehensive security analysis, it is essential to consider the feasibility of simple extraction and the resistance to attacks such as brute-force. For mobile cryptocurrency wallets, pre-rooting and physical access using tools like Android Debug Bridge (ADB) are essential for analysis. Therefore, security analysis research based on local files of desktop cryptocurrency wallets is needed, as data extraction is relatively easier. In this paper, we verified whether desktop cryptocurrency wallets securely store data and conducted a security analysis against brute-force attacks.

Table 1. Research trends on the security of cryptocurrency wallets based on local data.

Environment	Target Application	Data Source	Data Acquisition Window	Private Key Acquisition	Attack Execution	Refer
Mobile (Android)	Mycelium BitcoinWallet Bitpay Coinpayments	Local Disk	Always	Success	X [5]	[17]
	48 Apps	Both [1]	Varies [2]	Partial Success [3]	O [6]	[11]
	311 Apps	Both	Varies	Partial Success	O	[18]
Desktop (Windows)	Monero Verge	Both	Varies	Memory Only	X	[19]
	Multibit HD Electrum	Memory	Temporary	Depends on Conditions [4]	O	[20]
	Ethereum's Keystore File	Local Disk	Always	Depends on Conditions	O	[21]
	9 Apps	Both	Varies	Memory Only	X	[22]

[1] "Both" means that the analysis was performed using data obtained from both memory and the local disk. [2] "Varies" means that memory data are temporarily available, while local data are always accessible. [3] "Partial Success" means that the attack was carried out on multiple applications, but only some were successful. [4] "Depends on Conditions" refers to a study in which brute-force attacks were performed to obtain private keys or mnemonic codes. The success rates varied depending on the attack methods used. [5] "X" indicates that only the storage method, such as whether sensitive data is encrypted, was analyzed. [6] "O" signifies that attack experiments, such as dictionary attacks, were conducted to obtain sensitive data

3. Methodology for Security Analysis against Brute-Force Attacks on Password

In this section, we establish a methodology for analyzing the security of desktop cryptocurrency wallets against attacks utilizing local data storage:

- Phase 1: analysis of implementation flaws in wallets from a security perspective
- Phase 2: analysis of the wallets' resistance to brute-force attacks

3.1. Analysis of Implementation Flaws in Wallets from a Security Perspective

Desktop cryptocurrency wallets have to encrypt keys and data on user devices to prevent information leakage and subsequent wallet hacking [23]. Therefore, it is necessary

to inspect the implementation flaws in the wallets from a security perspective. Phases 1—(1), (2), and (3) in Figure 1 each show the following essential inspection points:

- Are wallet-related data stored securely?
- Is an appropriate password management policy used?
- Are sufficiently secure encryption algorithms used to protect the data storage?

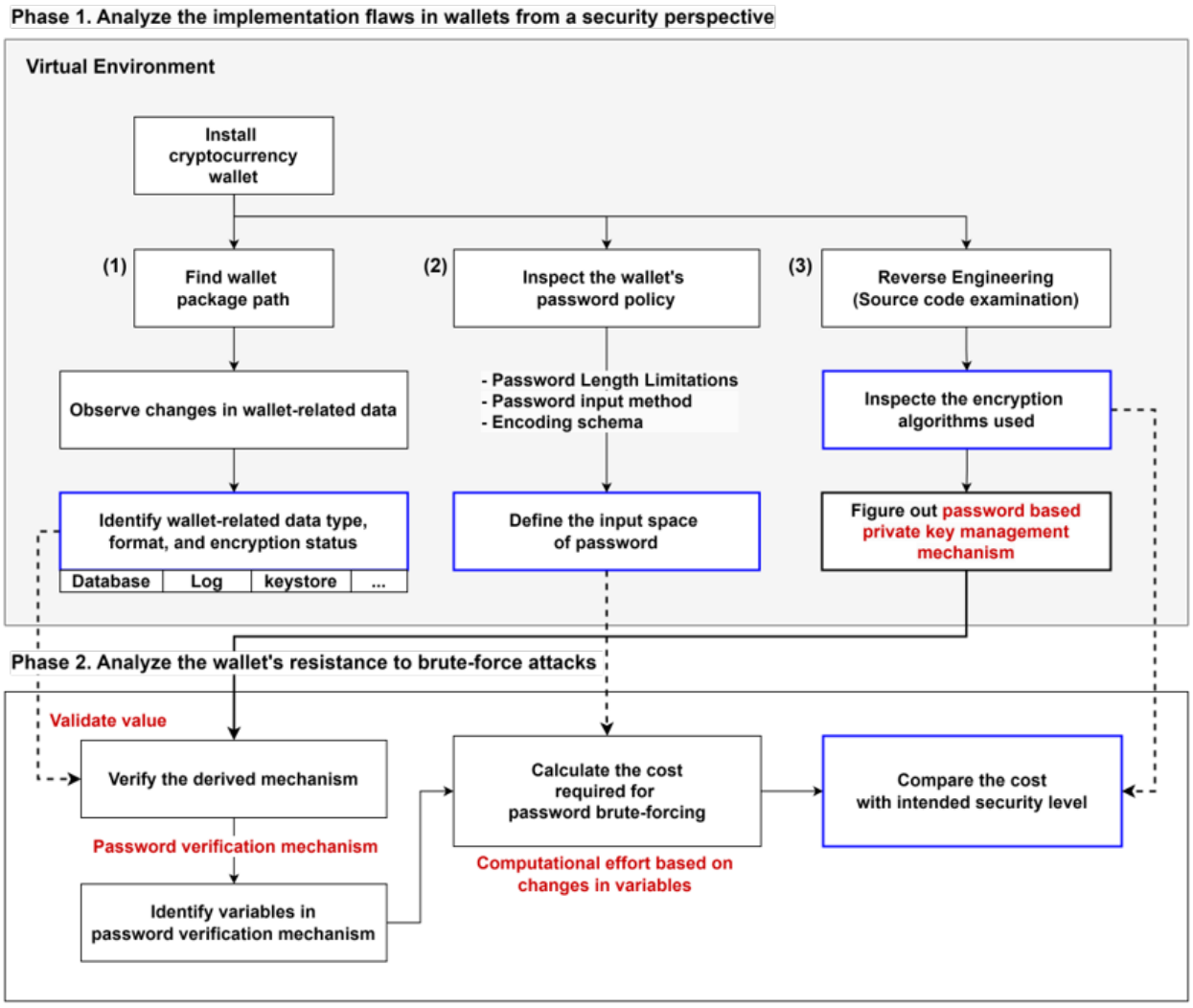

Figure 1. General methodology for analyzing the security of desktop cryptocurrency wallets against password brute-force attacks. The blue box represents the stage where answers to the questions necessary for analysis in each phase are derived.

First, locate the wallet's package path, and determine whether the data stored there are encrypted. The package path is where data related to application execution are stored and varies for each application. It is commonly located in the user's "AppData" directory or in the "Program Files" directory. After finding the package path, identify the types and forms of wallet-related data present and whether these data are encrypted. Wallet-related data can exist in various forms, such as databases or log files.

Next, inspect the target wallet's password policy. First, determine the minimum and maximum lengths of passwords that can be used. Then, analyze the size of the input space based on the password input method and encoding scheme. The encoding scheme commonly used for processing user input data is UTF-16. In this case, the input space for a single character in the password is 1,111,934 characters, approximately $2^{20.1}$, which excludes 2178 control characters [24]. Password input methods can be divided based on whether copy–paste is allowed. If copy–paste is not allowed, only 94 characters can be

entered via the keyboard (excluding the space). In this case, the input space for a single character in the password is approximately $2^{6.55}$.

Finally, analyze whether the wallet uses sufficiently secure algorithms to protect data storage. We need to identify the wallet's data encryption mechanism to perform this. Source code examination, storage data analysis, and reverse engineering can be performed.

3.2. Analysis of Wallet Resistance to Brute-Force Attacks

Desktop cryptocurrency wallets use a user-defined password for security. This section proposes a methodology to analyze resistance to brute-force attacks targeting passwords. The methodology is shown in Phase 2 in Figure 1.

First, a password verification oracle must be established to verify passwords. The password verification mechanism is derived from the data encryption mechanism analyzed in Section 3.1. In Phase 1—(1), we identify data within the wallet-related data that can be used as password validation values. If this value can be used to verify the password-based private key management mechanism derived in Phase 1—(3), the verification process is the password verification mechanism. Detailed examples of mechanism derivation can be found in Section 5. Additionally, variables and constants must be identified to implement an oracle that performs repetitive verification. For example, in the password verification process, IV and Salt are constants, the password is a variable, and HMAC is influenced by the variable.

Next, calculate the computational effort required for brute-forcing passwords. This involves identifying the input space of the internal variables of the oracle, utilizing the password input space analyzed in Phase 1—(2). Once the internal variables and their respective input spaces are identified, the computational effort required for a password brute-force attack can be calculated.

The computational effort can be compared with the intended security level of the cryptographic algorithm used to protect the data storage. If the calculated security is lower than the intended security, it is necessary to analyze which elements of each mechanism or variable have weakened the security.

The computational effort required for password brute-forcing can be represented using the input space size α, password length x, and time β taken for a cracking attempt. Since the computation for exhaustive search grows exponentially, computing cost C can be expressed as a (1)

$$C = e^{\alpha x} * e^{\beta} \tag{1}$$

4. Analysis of Cryptocurrency Wallet Implementation from a Security Perspective

In this section, we analyze the methods used by cryptocurrency wallets for data storage and the security policies they employ. The focus is particularly on password and private key management, which directly impact the security of digital assets. We targeted Sparrow, Etherwall, and Bither, which manage Bitcoin and Etherwall, the cryptocurrencies with the highest market share [25]. These three wallets have yet to be the subject of prior studies and are also not supported by well-known password-cracking tools like Hashcat [26], John the Ripper [27], Elcomsoft [28], and Passware [29]. The analysis results are limited to a Windows 10 64-bit Desktop environment, and we performed the analysis using data obtainable from local storage only. The detailed analysis environment is described in Table 2.

The package paths of the three wallets identified according to the step described in Section 3 are listed in Table 3. The package paths for each wallet will be denoted as [Sparrow], [Etherwall], and [Bither] below.

Table 2. Specification of analysis target and environment.

Environment	Windows10 64-bit
Target	Sparrow 1.7.9.3 Etherwall 3.0.3 Bither 1.4.8
Tool	H2DB 2.1.214 Python 3.9.10 Visual Studio 2019

Table 3. Cryptocurrency wallets' package directory.

Cryptocurrency Wallet	**Package Directory**
Sparrow	%AppData%\Roaming\Sparrow\
Etherwall	%AppData%\Ethereum\
Bither	%AppData%\Roaming\Bither\

4.1. Sparrow

Sparrow [30] is a Bitcoin wallet that supports Windows, Linux, and Mac. Sparrow uses a Hierarchical Deterministic (HD) structure [31] and employs a mnemonic code for generating the master seed. Sparrow uses UTF-16 encoding and allows copy–paste input. Therefore, the input space for a single character in the password is approximately $2^{20.1}$. Sparrow does not impose a minimum length limit on password input, so the minimum password length that can be set is one character. The computational effort based on password length is shown in Table 4.

Table 4. Computational cost according to password length (exponent values with a base of 2; value n in the table represents the exponent in 2^n).

Password Length	**Sparrow**	**Etherwall**	**Bither**
1	20.1	20.1	-
2	40.2	40.2	-
3	60.3	60.3	-
4	80.4	80.4	-
5	100.5	100.5	-
6	120.6	120.6	39.3
7	140.7	140.7	45.85
8	160.8	160.8	52.4

Sparrow's package path is "[Sparrow]\wallets". Sparrow stores wallet-related data in the "walletName.mv.db" database. The ".mv.db" extension belongs to the open-source database management system H2DB. If no password is set, the database is stored in plaintext, and all the data are accessible within the database. If the wallet password is set, the database itself is encrypted, and it is necessary to input the password to access the wallet. The database is encrypted based on the user's password. The process of encrypting the database containing sensitive data is shown in Figure 2a. The encryption algorithm used for H2DB encryption is AES-128-CBC, aiming for 128-bit security.

The NIST-recommended minimum password length is eight characters [32], and Sparrow achieves a security level of 160.8-bit against brute-force attacks at this length. However, the computational complexity of passwords with six characters or less is much lower than 128-bit. Therefore, depending on the user's password configuration, Sparrow's data protection mechanism can be less secure than intended.

The DB key is derived from the user password. Sparrow uses the Argon2id [33] algorithm to derive the AES key, and the specific parameters used in the calculations are fixed values, as shown in Table 5. These parameters are hardcoded in Sparrow's source code.

The salt is a 16-byte value extracted from the H2DB header, specifically from offset = 23 to offset = 39. The subsequent steps after DB key generation are also derived from the DB key. Ultimately, all encryption keys are derived from the user's password.

Table 5. Detailed parameters of the Argon2id algorithm used for deriving the database encryption key in Sparrow wallet.

Parameter	Value	Note
Hash length	32	hardcoded
Salt length	16	hardcoded
Salt	-	parsing
Iteration	10	hardcoded
Memory	256 × 1024	hardcoded
Parallelism	4	hardcoded

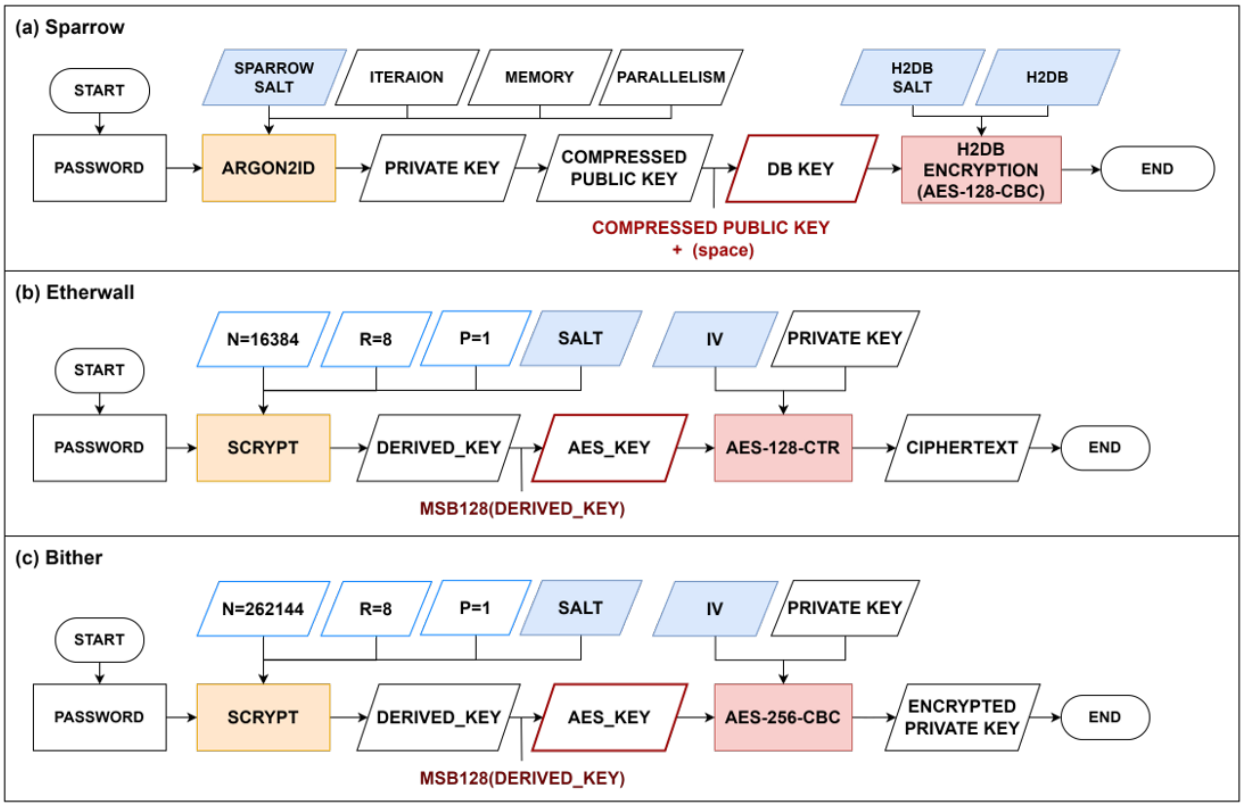

Figure 2. Private key encryption mechanisms of Sparrow, Etherwall, and Bither wallets.

4.2. Etherwall

Etherwall [34] is a wallet for Ethereum and supports Windows, macOS, and Linux. Etherwall is a non-HD [31] wallet, meaning a new password must be set each time an account is created. Each piece of account-related information, including the private key and transaction logs, is stored locally in a keystore file. Etherwall, like Sparrow, uses UTF-16 encoding, does not impose any constraints on the length of the password, and allows copy–paste.

Etherwall's keystore files are stored in the "[Etherwall]\keystore" directory. The file name follows the format of "[Account Creation UTC Time Data]-[0x[a-fA-F0-9]40]". [0x[a-fA-F0-9]40] is the account's address. Keystore files store the encrypted private key, the encryption algorithm used, the key derivation algorithm, and its parameters in plaintext. Etherwall utilizes AES and scrypt; the scrypt uses 262,144, 1, and 8 as N, P, and R, respectively. The nonce value, known as "salt", can also be found within the keystore file. Out of

the 32 bytes of output derived from scrypt, only the MSB 128 bits are used as the actual AES encryption key. The 16-byte Initialization Vector (IV) used in this process is also in the keystore file. The mechanism for encrypting the private key in Etherwall's keystore files is shown in Figure 2b. The encryption algorithm used to encrypt the private key is AES-128-CTR, intending 128-bit security. However, Etherwall's security is lower than intended when the password is six characters or less.

4.3. Bither

Bither [35] supports Bitcoin and is compatible with Windows, Linux, and macOS. All Bither addresses are disposable; users can manually create addresses as needed. Bither supports HD and non-HD wallets, requiring users to set a password. The mnemonic code used in HD wallets complies with the BIP39 standard [36], and non-HD wallets use entropy generated through random mouse movements to create private keys. Bither uses UTF-16 encoding for its password input, but copy–paste is prohibited. In this case, the password input space size is 94, approximately $2^{6.55}$. Bither limits the minimum length of a password to six characters. The results of calculating the required computational effort based on password length are shown in Table 4.

Information related to the Bither wallet is stored in the "address" directory located at the "[Bither]" path. The data are stored in an SQLite format database without a file extension. The wallet's sensitive data are stored in the "addresses" table of the "address.db". In the "encrypted_private_key" field of the addresses table, the encrypted private key, the IV, and the salt value are stored with "/" as a separator. The private key encryption mechanism is depicted in Figure 2c. Bither uses the scrypt [33] algorithm to derive the AES key. Scrypt's parameters N, P, and R are fixed in the source code, and each value is 16,384, 1, and 8, respectively. The algorithm employed for private key encryption is AES-256-CBC, which aims for 256-bit security. However, even when the password length exceeds eight characters [32], it does not meet the 256-bit security standard. Due to the small size of the input space, it does not even satisfy 128-bit security.

5. Security Analysis against Brute-Force Attack on Password

In this section, we analyze the security of each wallet against password brute-force attacks from a practical perspective. After constructing a password verification oracle for each wallet, experiments were conducted to deduce the brute-forcing cost based on password length.

5.1. Wallet Password Verification Oracle

Sparrow encrypts H2DB, which stores wallet-related data, and uses Argon2id for DB key derivation. Using the ODBC API [37], one can check the success of decryption through the returned value. All the necessary input values to derive the DB key and decrypt the encrypted DB are obtainable. We can implement an oracle for password verification using the API. The password verification oracle is depicted in Figure 3a.

Etherwall's private key is encrypted and stored in a keystore file. Etherwall independently verifies the password using the MAC value. Since the MAC value is also stored in plaintext, it can be used to verify the success of private key decryption. A MAC oracle can be implemented as all the necessary input values for MAC verification are obtainable through the keystore. The password verification mechanism is shown in Figure 3b. The MSB's 16 bytes of the derived key are concatenated with the 32 bytes of ciphertext to create the Hash Input value. Keccak-256 is then applied to the Hash Input, and the resulting digest is compared to the parsed MAC value to verify the password.

Bither's private key is encrypted and stored in a database, allowing retrieval of the encrypted private key. Internally, Bither uses key pair verification between the decrypted private key and the public key for password verification. However, we exploited Bither's PKCS#7 padding [38] in private key encryption to verify the password by examining the padding block. Through address.db, all the necessary input values for padding verification

are obtainable. We can implement the padding oracle, as shown in Figure 3c. Therefore, we can verify the password by confirming that the decrypted padding block matches the PKCS#7 format.

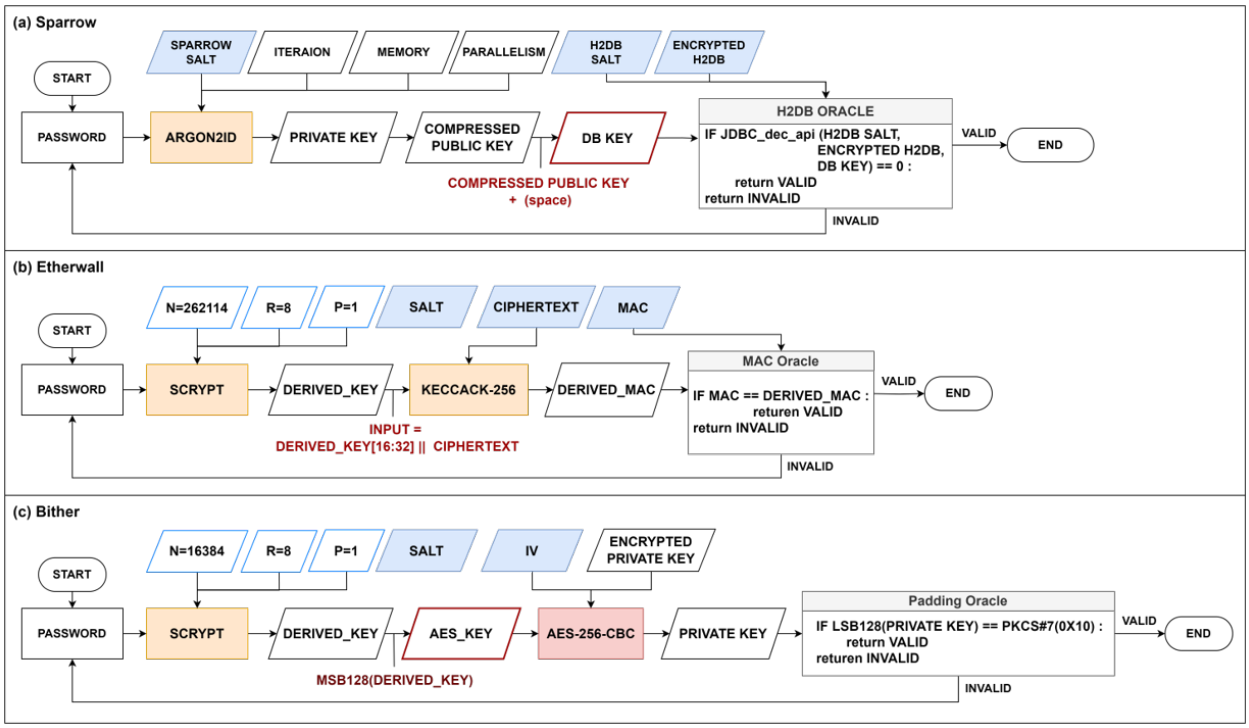

Figure 3. Password verification oracle of Sparrow, Etherwall, and Bither wallets.

5.2. Experimental Analysis from a Practical Perspective

We estimate the practical computing cost required to crack each cryptocurrency wallet through repeated experiments. The password input space is assumed to consist of 94 characters, which includes 62 alphanumeric characters (26 uppercase, 26 lowercase, and 10 digits) and 32 special characters, as shown in Table 6.

Table 6. Password input space.

Type	Characters	The Number
Alphabet	a–zA–Z	52
Number	0–9	10
Special Characters	‘!~@#$%&*()[]{};:’" <>,.?/=+−_	32

5.2.1. Password Sampling

The password-guessing method significantly affects cracking performance as the password length increases. To mitigate the performance differences due to sampling, we performed random sampling 1000 times for each password length and calculated the average cracking cost for each sample. We considered conducting more than 1000 repetitions adequate to ensure statistical significance. However, when the password length is one, the total number of samples is less than 1000, so we used the average cracking performance for 620 passwords. As the sample size is insufficient for lengths greater than or equal to 10, we considered password lengths from 1 to 9. The dictionaries used for password sampling were bt4-password [39], darkc0de [40], openwall.net-all [41], alleged-gmail-passwords [42], md5decryptor.uk [43], and rockyou [44]. The number of password samples extracted from these dictionaries by length is shown in Table 7.

Table 7. Number of samples according to the length of the password.

Password Length	Samples
1	620
2	9251
3	96,975
4	276,439
5	807,060
6	3,506,648
7	4,144,754
8	5,809,598
9	3,908,405
10	44

5.2.2. Experiment

The time required for a single cracking attempt measured after conducting repetitive cracking one million times for each wallet is presented in Table 8. We used the CPU clock as the measurement unit to ensure consistent performance regardless of the experimental environment. The method for calculating the average cracking cost is shown in Figure 4. Calculate the average cracking cost for 1000 random samplings, and repeat this process 1000 times to obtain the final average cost. For comparison, we also calculated the intended maximum security of the algorithms used in the wallet design. The average cracking cost for each algorithm represents the clock required to execute the algorithm once in the same environment. We used the OpenSSL [45] library for cryptographic operations, and all cracking costs may vary depending on the library and computing environment used. We used Intel i7-13700k [46].

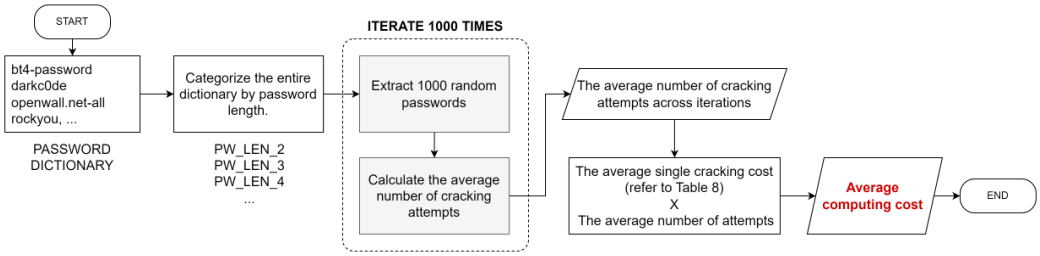

Figure 4. Method for calculating the average cracking cost.

Table 8. Average cracking cost.

Target	Average Cracking Cost (CPU Clock)
Sparrow	691.461
Etherwall	412.93
Bither	25.07
AES-128-CBC	0.000699
AES-128-CTR	0.000483
AES-256-CBC	0.000501

The experimental process is as follows:

1. Measure the average number of repetitions needed for a brute-force attack based on repetitive password sampling. Brute-forcing is performed sequentially from ASCII 33 to ASCII 126, and the number of repetitions for cracking is determined accordingly.
2. Calculate the average computing cost by multiplying the number of repetitions by the average cracking cost per clock.
3. For comparison with the intended security, calculate the ideal computing cost by multiplying the maximum number of cases required for a complete survey for each algorithm by the cracking cost.

5.2.3. Experiment Result

The estimated average computing cost obtained through steps 1 and 2 is presented in Table 9. By practically limiting the input space to keyboard inputs, the input space α is fixed at 1.97. The time for one cracking attempt β increases in the order of Sparrow, Etherwall, and Bither. According to (1), the regression equations derived based on the data for passwords of lengths 1 to 9 are as follows:

$$C_{\text{Sparrow}} = e^{1.97x} * e^{2.66} \tag{2}$$

$$C_{\text{Etherwall}} = e^{1.97x} * e^{2.43} \tag{3}$$

$$C_{\text{Bither}} = e^{1.97x} * e^{1.21} \tag{4}$$

Table 9. Average computing cost according to the length of the password.

Password Length	Sparrow	Etherwall	Bither
1	41,915	25,144	1519
2	4,041,900	2,419,944	144,520
3	3.98×10^8	2.38×10^8	1.43×10^7
4	3.61×10^{10}	2.16×10^{10}	1.33×10^9
5	3.36×10^{12}	2.00×10^{12}	1.21×10^{11}
6	2.96×10^{14}	1.77×10^{14}	1.04×10^{13}
7	2.84×10^{16}	1.70×10^{16}	1.02×10^{15}
8	2.74×10^{18}	1.64×10^{18}	1.00×10^{17}
9	2.66×10^{20}	1.58×10^{20}	9.53×10^{18}

Figure 5 shows the cost required to crack the passwords of the three wallets on a logarithmic scale. This graph is based on the data from Table 9. While the actual data increase exponentially, the growth rate is so large that we transformed the data to a base-10 logarithmic scale for effective comparison. The red dotted line at the top of the graph indicates the cost required to achieve 128-bit security cracking, which is generally considered ideal security from brute-force attacks.

Etherwall and Bither both use scrypt [33] for key derivation. However, Etherwall uses an N parameter that is 16-times larger, resulting in a higher required cost than Bither. Sparrow employs Argon2id [33] for key derivation, which requires significantly more time than scrypt. This makes Sparrow's cracking cost the highest among the three wallets. Nonetheless, there remains a considerable gap between Sparrow's cost and the cost associated with 128-bit security. A critical observation in the experiment is that the cost required for brute-force attacks increases exponentially with the password length. All the data in Tables 4 and 9 and Figure 5 demonstrate the substantial rise in cost as the password length increases.

In this experiment, we limited the input space to keyboard inputs for a realistic comparison. This may result in a lower cost than the actual. For example, according to Table 4, Sparrow can achieve 128-bit security with a password length of seven or more. However, in our experiment, as per (2), a password length of 17 is required for 128-bit security. For Bither, since the input space is limited to keyboard inputs, a password length of 37 or more is needed to achieve 128-bit security as per (4). This shows that both password length and input space size are crucial.

Therefore, password management policies should allow copying–pasting for password input to ensure a large enough input space. A minimum password length of at least eight characters should also be enforced. The experimental code for each cryptocurrency wallet's password verification and iterative testing can be referenced in [47].

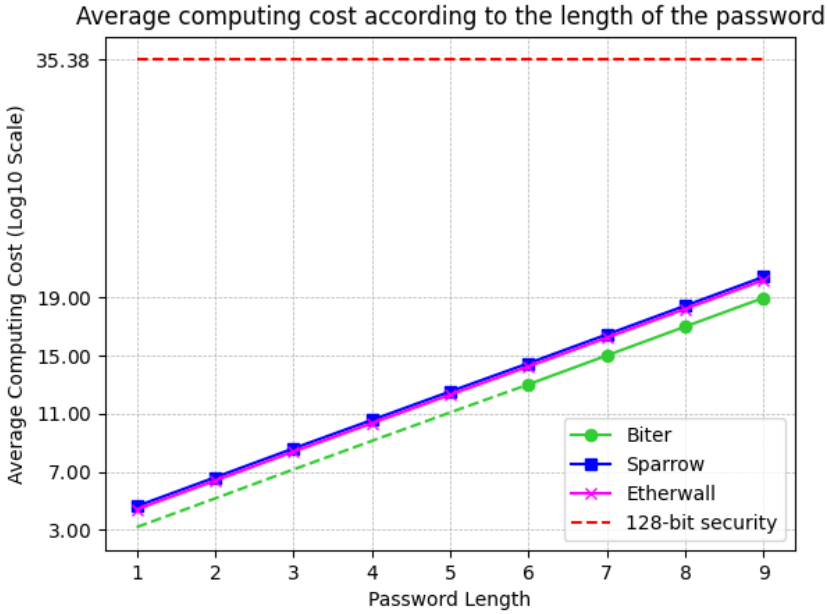

Figure 5. Average computing cost graph per password length.

6. Conclusions

As interest in cryptocurrencies grows and their value increases, the security of cryptocurrency wallets becomes even more critical. Password brute-force attacks can fundamentally occur on any cryptocurrency wallet that uses passwords. Therefore, wallet design must consider this threat and implement secure methods for storing sensitive data and managing passwords. In this paper, we examined the security of Sparrow, Etherwall, and Bither's resistance to password brute-force attacks based on local data and proposed practical guidelines for securely designing desktop cryptocurrency wallets. Our study is more practical than related research focused on memory analysis because it utilizes local data, making data acquisition easier.

Our investigation uncovered that all three wallets publicly retain sensitive parameters and nonce values used for private key encryption and need more password management policies. As a result, it is possible to implement a password verification oracle, making the security of the cryptocurrency wallets reliant on user passwords. Therefore, all wallets should enforce the NIST-recommended minimum password length of eight characters and allow copying–pasting of passwords to increase input space. Additionally, for Etherwall, it is recommended that keystore files containing all sensitive data be encrypted.

Our findings are particularly significant for open-source wallets with publicly available source code. Before deploying or selecting a desktop cryptocurrency wallet, the proposed analysis methodology in this paper can be used to evaluate the wallet's security. We expect our work to improve the overall security of cryptocurrency wallets. Additionally, the research findings can be utilized in investigations from a digital forensics perspective.

The research findings only cover desktop cryptocurrency wallets. Future research should expand the scope to analyze more cryptocurrency wallets across various operating systems, including mobile platforms. Additionally, it is necessary to perform security analysis from the memory perspective during program execution. The research can also include artifact analysis of wallet usage from a digital forensics perspective.

Author Contributions: Conceptualization, H.B. and B.S.; Methodology, H.B.; Software, H.B. and Y.J.; Validation, H.B., J.K. and Y.J.; Formal analysis, H.B.; Investigation, H.B. and J.K.; Resources, H.B. and J.K.; Data curation, H.B.; Writing—original draft, H.B.; Writing—review & editing, B.S. and S.G.; Supervision, B.S., S.G. and C.L.; Project administration, B.S.; Funding acquisition, C.L. All authors have read and agreed to the published version of the manuscript.

Funding: This study was supported by the Research Program funded by the SeoulTech (Seoul National University of Science and Technology), grant number 2021-0957.

Data Availability Statement: The data presented in this study are openly available in the 'Wallet_pwCrack_bruteforce' repository at https://github.com/mmbori/Wallet_pwCrack_bruteforce, reference number [47]. This repository contains the password verification code for each wallet used in our research.

Conflicts of Interest: The authors declare no conflicts of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of the data; in the writing of the manuscript; nor in the decision to publish the results.

References

1. Statista. *Statista Trend Report on the State of Cryptocurrency Adoption in 50 Countries Worldwide*; Statista: New York, NY, USA, 2023.
2. Nakamoto, S. Bitcoin: A Peer-to-Peer Electronic Cash System. Decentralized Business Review. 2008. Available online: https://assets.pubpub.org/d8wct41f/31611263538139.pdf (accessed on 8 May 2024).
3. CoinMarketCap. Available online: https://coinmarketcap.com/currencies/bitcoin/ (accessed on 17 May 2024).
4. Victor Manuel Portillo Rulz. DIARIO OFICIAL. 2021. Available online: https://cdn.inclusionfinanciera.gob.sv/wp-content/uploads/2021/06/Ley-Bitcoin.pdf (accessed on 12 May 2024).
5. Turner, S.; Brown, D.; Yiu, K.; Housley, R.; Polk, T. *RFC 5480—Elliptic Curve Cryptography Subject Public Key Information*; National Institute of Standards and Technology: Gaithersburg, MD, USA, 2009.
6. National Cyber Security Centre. Available online: https://www.ncsc.gov.uk/static-assets/documents/malware-analysis-reports/infamous-chisel/NCSC-MAR-Infamous-Chisel.pdf (accessed on 27 May 2024).
7. Kaspersky. Available online: https://www.kaspersky.com/resource-center/threats/mars-stealer-malware (accessed on 27 May 2024).
8. Chainalysis. The Chainalysis 2023 Crypto Crime Report. 2023. Available online: https://go.chainalysis.com/2023-crypto-crime-report.html (accessed on 19 October 2023).
9. Nazarov Alexander, Malanii Oleh. Atomic Wallet Hack: Overview xAnd Ongoing Investigation. 2023. Available online: https://hacken.io/discover/atomic-wallet-hack/ (accessed on 18 May 2024).
10. Alphapo Hot Wallets Hacked for over $31 Million. Available online: https://cointelegraph.com/news/alphapo-hot-wallets-hacked-for-over-31-million/ (accessed on 18 May 2024).
11. Sai, A.R.; Buckley, J.; Le Gear, A. Privacy and security analysis of cryptocurrency mobile applications. In Proceedings of the 2019 Fifth Conference on Mobile and Secure Services (MobiSecServ), Miami Beach, FL, USA, 2–3 March 2019; pp. 1–6
12. Houy, S.; Schmid, P.; Bartel, A. Security Aspects of Cryptocurrency Wallets—A Systematic Literature Review. *ACM Comput. Surv.* **2023**, *56*, 1–31. [CrossRef]
13. Nowroozi, E.; Seyedshoari, S.; Mekdad, Y.; Savaş, E.; Conti, M. Cryptocurrency wallets: Assessment and security. In *Blockchain for Cybersecurity in Cyber-Physical Systems*; Springer International Publishing: Cham, Switzerland, 2023.
14. Eskandari, S.; Clark, J.; Barrera, D.; Stobert, E. A first look at the usability of bitcoin key management. *arXiv* **2018**, arXiv:1802.04351.
15. Melicher, W.; Ur, B.; Segreti, S.M.; Komanduri, S.; Bauer, L.; Christin, N.; Cranor, L.F. Fast, lean, and accurate: Modeling password guessability using neural networks. In Proceedings of the 25th USENIX Security Symposium (USENIX Security 16), Austin, TX, USA, 10–12 August 2016.
16. Microsoft, BitLocker Overview. 2023. Available online: https://learn.microsoft.com/en-us/windows/security/operating-system-security/data-protection/bitlocker/ (accessed on 7 May 2024).
17. Haigh, T.; Breitinger, F.; Baggili, I. If I had a million cryptos: Cryptowallet application analysis and a trojan proof-of-concept. In Proceedings of the Digital Forensics and Cyber Crime: 10th International EAI Conference, ICDF2C 2018, New Orleans, LA, USA, 10–12 September 2018; pp. 45–65.
18. Uddin, M.S.; Mannan, M.; Youssef, A. Horus: A security assessment framework for android crypto wallets. In Proceedings of the Security and Privacy in Communication Networks: 17th EAI International Conference, SecureComm 2021, Virtual Event, 6–9 September 2021; Part II 17; pp. 120–139.
19. Koerhuis, W.; Kechadi, T.; Le-Khac, N.A. Forensic analysis of privacy-oriented cryptocurrencies. *Forensic Sci. Int. Digit. Investig.* **2020**, *33*, 200891. [CrossRef]
20. Volety, T.; Saini, S.; McGhin, T.; Liu, C.Z.; Choo, K.K. Cracking Bitcoin wallets: I want what you have in the wallets. *Future Gener. Comput. Syst.* **2019**, *91*, 136–143. [CrossRef]
21. Praitheeshan, P.; Xin, Y.W.; Pan, L.; Doss, R. Attainable hacks on Keystore files in Ethereum wallets—A systematic analysis. In Proceedings of the Future Network Systems and Security: 5th International Conference, FNSS 2019, Melbourne, VIC, Australia, 27–29 November 2019; pp. 99–117.
22. Zollner, S.; Choo, K.K.; Le-Khac, N.A. An automated live forensic and postmortem analysis tool for bitcoin on windows systems. *IEEE Access* **2019**, *7*, 158250–158263. [CrossRef]
23. Suratkar, S.; Shirole, M.; Bhirud, S. Cryptocurrency wallet: A review. In Proceedings of the 4th International Conference on Computer, Communication and Signal Processing (ICCCSP), Chennai, India, 28–29 September 2020.

24. Unicode. Available online: https://home.unicode.org/ (accessed on 3 May 2024).
25. Statista, Dominance of Bitcoin and other Crypto in the Overall Market from 2nd Quarter of 2013 to 3rd Quarter of 2023. 2023. Available online: https://www.statista.com/statistics/730782/cryptocurrencies-market-capitalization/ (accessed on 11 May 2024).
26. Hashcat. Generic Hash Types. Available online: https://hashcat.net/wiki/doku.php?id=example_hashes (accessed on 3 May 2024).
27. Openwall. John the Ripper Password Cracker. Available online: https://www.openwall.com/john/ (accessed on 3 May 2024).
28. Elcomsoft. Elcomsoft Distributed Password Recovery—Supported Document Formats. Available online: https://www.elcomsoft.com/help/en/edpr/\server/1_3_.html (accessed on 3 May 2024).
29. Passware. Passware Kit Forensic 2024 V1 Supports 350+ File Types. Available online: https://www.passware.com/kit-forensic/filetypes/ (accessed on 3 May 2024).
30. Sparrow. Available online: https://github.com/sparrowwallet/sparrow (accessed on 15 April 2024).
31. bip-0032.mediawiki. Available online: https://github.com/bitcoin/bips/blob/master/bip-0032.mediawiki (accessed on 8 May 2024).
32. Grassi, P.A.; Fenton, J.L.; Newton, E.M.; Perlner, R.A.; Regenscheid, A.R.; Burr, W.E.; Richer, J.P.; Lefkovitz, N.B.; Danker, J.M.; Choong, Y.-Y.; et al. *NIST Special Publication 800-63B*; National Institute of Standards and Technology: Gaithersburg, MD, USA, 2023.
33. Turan, M.S.; Barker, E.; Burr, W.; Chen, L. *Recommendation for Password-Based Key Derivation: Part 1: Storage Applications*; Sp 800-132; National Institute of Standards and Technology: Gaithersburg, MD, USA, 2010.
34. Etherwall. Available online: https://github.com/almindor/etherwall (accessed on 15 April 2024).
35. Bitherj. Available online: https://github.com/bither/bitherj (accessed on 15 April 2024).
36. bip-0039-wordlists.md. Available online: https://github.com/bitcoin/bips/blob/master/bip-0039/bip-0039-wordlists.md (accessed on 9 May 2024).
37. SQLDriverConnect. Available online: https://learn.microsoft.com/ko-kr/sql/odbc/reference/syntax/sqldriverconnect-function?view=sql-server-ver16 (accessed on 29 April 2024).
38. PKCS Padding Method. Available online: https://www.ibm.com/docs/en/zos/2.4.0?topic=rules-pkcs-padding-method (accessed on 4 May 2024).
39. bt4-password.txt. Available online: https://github.com/danielmiessler/SecLists/blob/master/Passwords/bt4-password.txt (accessed on 4 May 2024).
40. darkc0de.txt. Available online: https://github.com/danielmiessler/SecLists/blob/master/Passwords/darkc0de.txt (accessed on 4 May 2024).
41. openwall.net-all.txt. Available online: https://github.com/danielmiessler/SecLists/blob/master/Passwords/openwall.net-all.txt (accessed on 4 May 2024).
42. alleged-gmail-passwords.txt. Available online: https://github.com/danielmiessler/SecLists/blob/master/Passwords/Leaked-Databases/alleged-gmail-passwords.txt (accessed on 4 May 2024).
43. md5decryptor-uk.txt. Available online: https://github.com/danielmiessler/SecLists/blob/master/Passwords/Leaked-Databases/md5decryptor-uk.txt (accessed on 4 May 2024).
44. rockyou.txt.tar.gz. Available online: https://github.com/danielmiessler/SecLists/blob/master/Passwords/Leaked-Databases/rockyou.txt.tar.gz (accessed on 4 May 2024).
45. OpenSSL. Available online: https://www.openssl.org/ (accessed on 24 April 2024).
46. Intel Core i7-13700K Processor. Available online: https://www.intel.com/content/www/us/en/products/sku/230500/intel-core-i713700k-processor-30m-cache-up-to-5-40-ghz/specifications.html (accessed on 2 May 2024).
47. Wallet_pwCrack_bruteforce. Available online: https://github.com/mmbori/Wallet_pwCrack_bruteforce (accessed on 14 April 2024).