Short Paper: Oops. . . I Did It Again. I Reused my Nonce.

Vincent Jacquot[0009−0007−8026−5277], Benoit Donnet[0000−0002−0651−3398]⋆

Université de Liège, Montefiore Institute, Liège, Belgium

Abstract. The Elliptic Curve Digital Signature Algorithm (ECDSA) is widely used by cryptocurrencies to authenticate transactions through signature. Signing with ECDSA involve the generation of a nonce, a number that must be used only once and randomly regenerated between each signature. This is particularly important as it has been demonstrated that reusing the nonce between signatures allows attackers to recover private keys and, ultimately, steal funds.
This paper presents a large-scale analysis of ECDSA signatures across several blockchains, identifying numerous cases of nonce reuse within blockchains (*within-reuse*) but also between different blockchains (*cross-reuse*), leading to a larger attack surface. Our approach combines cryptographic techniques for private key recovery, the deployment of honeypots to collect ground-truth evidence, and the analysis of real-world incidents to better understand the scope of the problem. Notably, we recovered 3,620 private keys across several blockchains, revealing both cross-reuse. Our analysis estimates the potential financial loss could reach up to 101 million EUR.

1 Introduction

Many of the most popular cryptocurrencies use the *Elliptic Curve Digital Signature Algorithm* (ECDSA) [1], an asymmetric cryptographic algorithm, to generate digital signatures. ECDSA requires the secure generation of *nonces*: random values that must be used only once. Reusing a nonce can allow an adversary to recover the associated private key, leading to potential theft of funds. The vulnerabilities were already studied in previous papers [2–6].

This paper investigates real-world vulnerabilities arising from nonce reuse across six cryptocurrencies. We systematically analyzed their complete transaction history to evaluate the security impact of reused nonces. To conduct this study, we deployed honeypots, which involve depositing funds into cryptocurrency addresses that have reused nonces, to obtain ground-truth evidence and analyzed empirical cases of nonce reuse.

Compared to prior work, this paper introduces the following novel contributions: (*i*) it presents the first systematic study of ECDSA nonce reuse on Bitcoin

⋆ This work is supported by the CyberExcellence project funded by the Walloon Region, under number 2110186. The authors also extend our thanks to TRM Labs for providing access to raw blockchain data.

Cash, Litecoin, Dogecoin, and Dash; (*ii*) it shows that nonce reuse can occur within a single blockchain (*within-reuse*), but also across several (*cross-reuse*); (*iii*) it provides the first confirmation that attackers are actively exploiting nonce reuse vulnerabilities on both Ethereum and Bitcoin; Finally, we publicy release the tool we have developed and employed to derive the private keys from a dataset of signatures (`github.com`).

Moreover, in line with prior work, our contributions also include (*iv*) estimating the financial impact of attacks enabled by nonce reuse, with a potential loss of €101 million; (*v*) analyzing attackers' behavior and presenting signs of their activity across the full transaction history of Bitcoin and Ethereum; and (*vi*) discussing potential mitigation strategies to prevent similar vulnerabilities going forward.

2 Related work

The risk of reusing a nonce with the same private key is well known, in particular on Bitcoin [2–6].

Bos et al. [2] ran an analysis on a 2013 Bitcoin blockchain snapshot (up to block number 252,450). They managed to recover 158 private keys. They were able to identify three keys belonging to `Bitcoincard` and several `Blockchain-.info` accounts.

Moreover, Brengel and Rossow [3] extended the attack by including nonce reuse across distinct private keys, modelling signatures in a bipartite graph to detect vulnerable private keys. This is the technique adopted in this paper. They ran the attack on a snapshot of Bitcoin (up to block 506,071) and managed to recover 2,537 private keys. They also provided a calibrated estimate of the potential financial loss from these attacks (up to 412.80₿).Particular nonces such as the most frequent one which appears 2 million times or special values which were likely chosen manually by humans were discussed. Finally, they performed a similar temporal analysis as we do, observing five spikes between February 2013 and September 2015 that align in both time and magnitude with those highlighted in our paper.

Breitner and Heninger [4] focused on the vulnerability introduced by the use of weak nonces. They used a lattice-based approach to recover private keys, which is not considered in this work. They performed their analysis on a 2018 snapshot of Bitcoin (up to block 541,244) and Ethereum (up to block 6,346,730). They also analysed a portion of the Ripple blockchain. As a side effect of their work, they recovered 1,296 Bitcoin, three Ethereum, and one Ripple private keys from repeated nonces, in addition to the private keys their lattice-based approach can extract but ours cannot.

Macchetti [6] introduced a nonce-related attack that exploits algebraic recurrence relations in certain nonce-generation procedures, allowing key recovery from only a few signatures. Their results intersect with ours since a repeated nonce corresponds to the simplest recurrence relation covered by their framework. However, their method appears unable to recover more private keys than the approach used in this paper.

3 Background

The *Bitcoin* (Btc) is a protocol whose history starts in 2008 [7] and defines the first decentralized digital currency, the *bitcoin* (₿). This innovation paved the way for many other cryptocurrencies, such as the Ethereum project [8], which also allows for the exchange of another cryptocurrency called *ether* (ETH).

Many cryptocurrencies' codebases are forks of Bitcoin: Litecoin (Ltc) [9], Dogecoin (Doge) [10], Dash (Dash) [11], and BitcoinCash (Bch) [12]. In the following, we will refer to Bitcoin and its forks collectively as Bitcoin et al.

Bitcoin et al. and Ethereum rely on ecdsa over the SECP256k1 curve [13,14] to authenticate messages such as financial transactions. The curve is defined by several parameters, such as the generator point G and the order n of G.

To sign a message with ecdsa, end-users generate a random integer, d_A, which serves as the private key. The public key Q_A can then be derived mathematically from this private key. Note that this process is a one-way function; it is not possible to retrieve d_A from Q_A. Then, the private key can be used to sign any number of messages [15] as follows:

Step 1: Compute the digest e of the message m as $e = HASH(m)$, where $HASH$ is a cryptographic hash function.
Step 2: Let z be the L_n leftmost bits of e, where L_n is the bit length of the group order n.
Step 3: Select a cryptographically secure random integer $k \in [1, n-1]$, the *nonce*.
Step 4: Calculate the curve point $(x_1, y_1) = kG$.
Step 5: Calculate $r = x_1 \ mod \ n$. If $r = 0$, go back to Step 3.
Step 6: Calculate $s = k^{-1}(z + rd_A) \ mod \ n$. If $s = 0$, go back to Step 3.
Step 7: The signature is the pair (r, s). And $(r, -s \ mod \ n)$ is also a valid signature.

Finally, blockchains employ the concept of *cryptocurrency address*: a unique alphanumeric string that represents a destination for sending or receiving cryptocurrency. Address generation schemes typically rely on the owner's public key. On Ethereum, each public key maps to a single address, whereas on Bitcoin et al., a single public key can generate multiple distinct addresses. Note that Ethereum enables the deployment of semi-autonomous pieces of code known as *smart contracts*. Those smart contracts are also identified by an address, but their behavior is totally predefined by their code.

4 Data Collection

We formalize a transcript as a tuple (r, s, pk, z, ts), where r and s together represent the signature, pk is the public key, z is the message digest, and ts is the timestamp of the signature. We extracted signatures from six blockchains: Btc, Bch, Doge, Ltc, Dash, Eth. This dataset spans the entire lifetime of these blockchains up to June 30th, 2024. The exact blockchain heights considered in this work are listed in the long version of the paper [16].

Our industrial partner, TRM Labs, provided us with access to raw blockchain data, such as transactions for Bitcoin et al. Since the mechanism for verifying signatures is well-documented and standardized [17–19], extracting r, s, pk from raw transactions is straightforward. However, producing the transaction digest is more complex [20, 21]. For this, we rely on a Python implementation [22] to generate Bitcoin transaction digests and adapted this software to generate digests on Bitcoin-like blockchains.

For Ethereum, we extracted signatures thanks to an archive node running Lighthouse [23] and Erigon [24]. First, we extracted all signatures used to authenticate the transactions. Every execution client must implement the same JSON-RPC specification, which allows for the retrieval of the r, s values, and the sender's address for any transaction [25]. We relied on another library, `eth-account` [26], to compute the message digest from the raw transaction data.

Finally, it is possible to retrieve the public key pk from the message digest and the signature [15]. For SECP256k1 [27], public key recovery can yield up to four possible public keys. Since an Ethereum address is formed of the last 20 bytes of the public key hash, we can accurately identify the correct public key.

Ethereum provides a precompiled contract at address 0x01 that performs signature verification. On-chain signatures are sent as transaction parameters to this contract and constitute our second source of Ethereum signatures.

5 Attack

5.1 Single-Key Nonce Reuse

With ECDSA, using the same nonce to produce two signatures with the same private key allows anyone to retrieve the private key. From Step 4 and Step 5, it is clear that both signatures will share the same first half of the signature, i.e., r. Let s_1 and s_2 be the respective second part of both signatures, we obtain a system of two linearly independent equations with two unknowns: the shared nonce k and the private key d_A. Thus, the system is uniquely solvable, and the private key can be retrieved as follows:

$$\begin{cases} s_1 = k^{-1}(z_1 + rd_A) \ mod \ n \\ s_2 = k^{-1}(z_2 + rd_A) \ mod \ n \end{cases} \Leftrightarrow \begin{cases} k = (s_1 - s_2)^{-1}(z_1 - z_2) \ mod \ n \\ d_A = r^{-1}(s_1 \cdot k - z_1) \ mod \ n \end{cases}$$

As stated in Step 6, for every signature (r, s), $(r, -s)$ is also a valid signature. Hence, the user may have published $-s_1$ and/or $-s_2$ instead of s_1 and s_2. To address this issue, we can simply brute-force all possibilities, four in this case, until the expected private key is recovered.

5.2 Multiple Keys Nonce Reuse

Instead of focusing on nonces reused by the same key, private keys can also be recovered when nonces are reused across distinct private keys. For example, consider two private keys d_1 and d_2 used with two nonces k_1 and k_2. The resulting system of four linearly independent equations in the four unknowns (k_1, k_2, d_1, d_2) is uniquely solvable.

$$\begin{cases} s_1 = k_1^{-1}(z_1 + r_1 d_1) \ mod \ n \\ s_2 = k_2^{-1}(z_2 + r_2 d_1) \ mod \ n \\ s_3 = k_1^{-1}(z_3 + r_1 d_2) \ mod \ n \\ s_4 = k_2^{-1}(z_4 + r_2 d_2) \ mod \ n \end{cases} \tag{1}$$

Moreover, this principle can be extended. To identify such solvable systems of equations, one can rely on graph theory as originally framed by Brengel and Rossow [3]. One can build a bipartite graph $G = (V_{pk} \cup V_r, E)$, where V_{pk} is the set of public keys and V_r the set of r values. An edge exists between a public key node and a r node if there exists a signature for this public key pk and r value. This representation directly exposes repeated nonce usage across multiple private keys. Any non-trivial cycle of $2{\cdot}n$ edges (i.e. signatures) translates into $2{\cdot}n$ linearly independent equations involving n private keys and n nonces, yielding $2 \cdot n$ unknowns.

5.3 Methodology

From the set of signatures, our tool selects a subset composed of signatures whose r values were used more than once. Then, it builds a bipartite graph from these signatures.

It first recovers private keys that were used to sign multiple messages with the same nonce, as described in Sec. 5.1. Then, the tool constructs systems of linear equations from signatures that form cycles, as described in Sec. 5.2.

Finally, the attack can be extended further as described by Brengel and Rossow [3]. From the set of recovered nonces, it is possible to detect all signatures that used one of the recovered nonce, as r is derived from the nonce (see Step 3 and Step 4 in Sec. 3), and to recover the private key. The same applies for the set of recovered private keys. We can generate the corresponding public keys from this set, fetch the corresponding signatures, and recover the nonces. This final step is repeated until no new private key and nonce is recovered.

6 Results

We initially applied our methodology separately to each blockchain. The number of private keys recovered for each is also shown in Table 1. Interestingly, the total number of distinct recovered private keys is not $3,608$ as expected but $3,605$. This discrepancy arises from three private keys being used across multiple blockchains.

Table 1: Results of the attack applied separately on every chain (within-reuse).

	Reused nonce data			**Attacks**	
	Sig.	**'r'**	**PubKey**	**PrivKey**	**Nonces**
Btc	2.50M	2441	5372	3,278	1696
Eth	21.3k	461	20.9k	119	102
Bch	141	7	8	6	6
Ltc	26	7	15	9	5
Doge	578	268	332	195	200
Dash	11	2	3	1	1
Total	**2.90M**	**3174**	**26.6k**	**3,605**	**2002**

Applying the attack on the combined set of signatures from all blockchains allows us to recover 3,620 private keys (cross-reuse). This indicates the presence of nonces being reused across multiple blockchains. At their peak, slightly less than 2,000€could have been stolen from the additional 15 private keys. Running the attack on all blockchains at once also allows us to crack five keys earlier than would have been possible with the attack being run on a per chain basis. For three of them, the key could be cracked 41 days earlier, and for the other two, around 20 minutes earlier.

6.1 Financial Loss Estimation

An important aspect is evaluating the potential of these attacks. All amounts in euros are based on cryptocurrency prices as of November 25th, 2024 throughout the paper. Fig. 1 illustrates the cumulative amount of vulnerable money over time. First, we observe nine occurrences over time where more than a million euros in cryptocurrencies could have been stolen. Most of these occurrences happened between 2014 and early 2016, with vulnerable Bitcoin keys being the main contributors. At its peak in December 2014, cryptocurrencies worth €25M could have been stolen. For the last two occurrences, in October 2022 and February 2023, vulnerable keys could have been exploited to steal €11.2M and €1.2M, respectively, with most of these amounts originating from Ethereum. Notably, the vulnerable addresses held cryptocurrencies worth €101M at their peak balances. The vast majority of the funds originate from Btc and Eth.

From Fig. 2, we can observe that 67.25% of the vulnerable addresses held cryptocurrencies worth less than a hundred euros. We observe the presence of addresses with large balances (more than €1M) that remained vulnerable for extended periods (more than 24 hours). This set includes two Ethereum addresses that were active in 2022 and 2023, and three Bitcoin addresses with peak balances in 2013, 2014, and early 2016.

6.2 Confirming Attackers Activity

Since the true owners of addresses are generally unknown and users can generate an unlimited number of addresses, a transfer from a vulnerable address

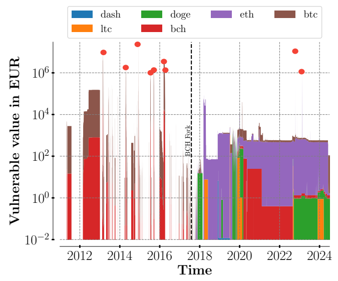

Fig. 1: Vulnerable value over time.

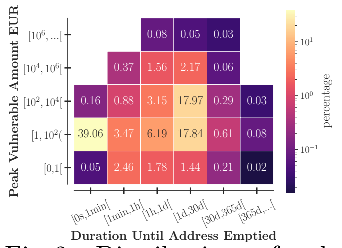

Fig. 2: Distribution of address balances and vulnerability duration.

does not necessarily indicate an attack. In this section, we focus on demonstrating that attackers are actively scanning BTC and ETH by deploying and monitoring honeypots. We deployed two types of honeypots: SKNR (Single-Key Nonce Reuse) and MKNR (Multiple Key Nonce Reuse) which respectively test the presence of attacks described in Sec. 5.1 and 5.2 on April 28th, 2025. All vulnerable addresses were funded with between 3 and 5 USD worth of cryptocurrency at that time.

SKNR honeypots consist of two funded addresses: the first address's private key uses the same nonce twice, and the second address's private key uses that nonce one. Hence, both keys are vulnerable. We also deployed a special honeypot where the same private key was used twice with the same nonce to produce one signature on ETH and one signature on BTC. This allows us to detect attackers monitoring multiple chains. MKNR honeypots involve three private keys and two nonces. The first two keys use every nonce exactly once, forming a solvable system of equations. The third private key uses the first nonce as well.

Finally, we also monitored the BTC mempool (the space where unconfirmed transactions reside before being included into the blockchain) with the Bitcoin Core RPC API. This method provides a convenient way to observe concurrent transactions from different attackers attempting to withdraw funds from the same honeypot.

BTC: We observed transactions in the mempool targeting our SKNR honeypots at the second the addresses became vulnerable. However, only the addresses which directly reused the same nonce were attacked. This suggests that some actors run automated attacks on BTC but limited to recovering a private key that reuse the same nonce. Finally, on the May 11th 2025, another attack siphoned all remaining untouched BTC addresses. Interestingly, we observed two distinct attempts to spend funds from a honeypot, suggesting that multiple attackers might be competing with each other. No transaction shows any sign of boosted fees.

However, these observations seem to conflict with the continued presence of vulnerable bitcoins, as illustrated in Fig. 1. As previously discussed, BTC supports multiple address formats, and most of the currently vulnerable funds are stored at an address using an uncommon format. Due to the nature of cryptocurrencies, it is not possible to contact the owners of the vulnerable addresses.

This may reinforce the intuition that attackers are often disorganized and lack a systematic approach.

Eth: Every vulnerable address was siphoned, suggesting that attackers are more methodical. We also observed that it took approximately three hours for the honeypot to be attacked and the funds transferred to the same address. The transactions' fees are not outliers compared to other transactions in the same block; however, transactions use the legacy version type.

Btc & Eth: As of December 2025, none of the vulnerable addresses were attacked, hence we can assume that no attackers were monitoring for cross-chain nonce reuse during this period of time across Btc and Eth.

6.3 Evolution of Attacker Activity

Bitcoin et al. We are also interested in determining whether attackers are consistently monitoring for vulnerabilities. Bitcoin et al. provide a perfect playground to answer this question. We focused on addresses vulnerable due to single-key nonce reuse, as it appears that attackers might not be employing more advanced attacks, as discussed in the previous subsection. For every money deposit made after the address became vulnerable, we measured the time it took for the funds to be transferred to another address. Fig. 3 presents the average time per quarter.

First of all, it appears that Btc and Bch experienced periods during which vulnerable deposits were systematically transferred to other addresses, though this does not seem to be the consistent trend over time. The earliest period dates back to the fourth quarter of 2015 for Btc. While we lack definitive proof that these transactions were attacks, this may suggest that attackers carry out such actions sporadically rather than consistently. For the other three blockchains, we have fewer data points, but the available evidence suggests that these chains might not currently be the targets of attacks. This might be explained by their lower popularity and value.

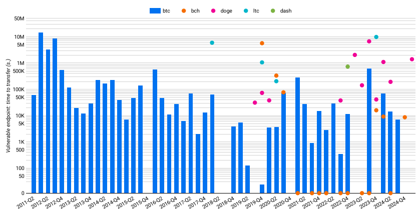

Fig. 3: Average time in seconds until a vulnerable deposit is transferred.

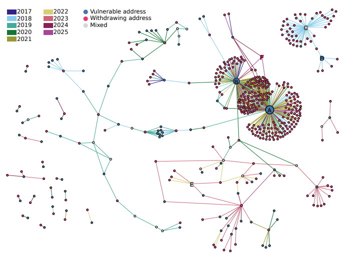

Fig. 4: Interaction between vulnerable and withdrawing addresses on Eth.

Eth As the transaction model differs totally from the other chains, we took another approach. We listed all addresses which withdrew funds from vulnerable addresses after they became vulnerable. We exclude from this list the addresses which received funds from vulnerable addresses before their vulnerability and the addresses which send funds to those vulnerable addresses. This allowed us to build the graph in Fig. 4. Each node represents an address (categorized as vulnerable, withdrawing, or mixed), and an edge is drawn when a fund transfer occurs between two addresses. We have investigated in detail the history of five addresses labeled A, B, and E.

A & B: Both addresses are publicly listed on websites that catalog known vulnerable keys[1]. Remarkably, these two addresses continue to receive funds, which are typically withdrawn shortly afterward. Furthermore, although the addresses were already vulnerable as early as September 2017, suspicious withdrawals only began approximately two months later.

E: This address first interacted with and withdrew funds from another vulnerable address in August 2022. In October 2022, E itself became vulnerable. Thirteen hours later, a suspicious transfer of 62 ether occurred. The withdrawing address forwarded the funds to the well-known mixer, TornadoCash. Later in October, a second withdrawing address stole 2 ether just 5 hours after it was deposited. In November, two more deposits of 6 and then 350 ether were claimed by yet another address, only 12 seconds after being made. This sequence of events strongly suggests that address E was indeed the victim of an attack in 2022.

Based on this sample of analyzed addresses, withdrawals from vulnerable Eth addresses date back as early as 2017, and we have strong evidence that at least one attack occurred in 2022. This hypothesis is further supported by the existence of a blog post already reporting such attacks in December 2021 [28].

7 Mitigation

Repeated nonces likely occur for three reasons. First, the most frequently used nonce, which accounts for 2.48M BTC transactions, generates a r of 21 bytes. As users pay their fee per byte, this special nonce allows them to decrease the transaction cost on Bitcoin et al. chains. The second most used nonce generates a 30-byte r. The financial argument becomes clear when considering the nature of UTXO blockchains. Second, improperly implemented software might be responsible. Two known examples are the Android bug in 2013 [29], or the bug in a JavaScript client's random number generator not being seeded correctly [30]. Finally, we have noticed nonces that might have been manually chosen by humans, e.g., 123456, 12345678, 0x01010...10101.

RFC 6979 [31] defines a deterministic nonce generation procedure that completely eliminates the need for a pseudo-random number generator. If this process were systematically used, repeated nonces would not occur, and the attack described in this paper would not be applicable.

A core reason these issues arise today is the lack of education on the topic. We aim to raise developer awareness of such issues. As a concrete action, we have

[1] Examples: `https://privatekeys.directory/keys?page=1&coin=eth` and `https://ethkey.net/`

developed two CTF levels for Ethernaut[2], a popular online Ethereum smart contract CTF challenge. Both levels highlight common vulnerabilties related to ECDSA, with one specifically focuses on repeated nonces. Moreover, publishing our tool to derive private keys from a dataset of signatures may assist companies.

8 Conclusion

Our analysis is the first to examine nonce reuse across the entire ETH, DOGE, DASH, LTC, and BCH blockchains. This study, which also includes BTC, is the first to investigate nonce reuse across multiple chains. It highlights that ECDSA nonce reuse remains a significant threat. In total, over 101M EUR in cryptocurrencies could have been stolen.

Through the use of honeypots, we have confirmed that attackers are actively targeting BTC and ETH as of April 2025. There are indications that these attacks are not carried out consistently, neither in terms of exploiting their full potential nor in their continuity over time. The earliest evidence of such activity for BTC and ETH dates back to 2015 and 2022, respectively.

Our investigation suggests that different actors may be involved. For BTC, we observe signs of automation, though the automated attacks appear suboptimal. Furthermore, we have gathered evidence that multiple actors are competing against each other. In contrast, the actor observed on ETH exhibited a more methodical approach, though the delayed response suggests that the attack is not fully automated. Additionally, our honeypot which requires attackers to monitor repeated nonces on BTC and ETH, demonstrates that attackers tend to focus on one blockchain at a time. Our findings also show that monitoring multiple blockchains would enable attackers to retrieve keys ahead of their competitors, ultimately allowing them to access more private keys.

We also present evidence that DOGE, DASH, and LTC are likely not under active scrutiny by attackers, possibly due to their lower value or popularity.

By shedding light on these issues, we hope this study will serve as a wake-up call for blockchain users and wallet developers. Without stronger safeguards, these vulnerabilities could continue to undermine trust in decentralized financial systems.

Ethical Considerations

Honeypots were deployed specifically to study malicious behavior, not to deceive or harm regular users. To verify our findings, we derived the private keys of vulnerable addresses. These keys were never shared with any third party, nor were they used to withdraw funds from those addresses.

[2] Refer to CTF levels 35 and 37 available at `https://ethernaut.openzeppelin.com`

References

1. D. Johnson, A. Menezes, and S. Vanstone, “The elliptic curve digital signature algorithm (ECDSA),” *International Journal of Information Security (IJIS)*, vol. 1, no. 1, pp. 36–63, August 2001.
2. J. W. Bos, J. A. Halderman, N. Heninger, J. Moore, M. Naehrig, and E. Wustrow, “Elliptic curve cryptography in practice,” in *Proc. Financial Cryptography and Data Security (FC)*, March 2014.
3. M. Brengel and C. Rossow, “Identifying key leakage of bitcoin users,” in *Proc. Research in Attacks, Intrusions, and Defenses (RAID)*, September 2018.
4. J. Breitner and N. Heninger, “Biased nonce sense: Lattice attacks against weak ECDSA signatures in cryptocurrencies,” in *Proc. Financial Cryptography and Data Security (FC)*, September 2019.
5. J. Ko and J. Kwak, “Private key recovery on bitcoin with duplicated signatures,” *KSII Transactions on Internet and Information Systems*, vol. 14, no. 3, pp. 1280–1300, March 2020.
6. M. Macchetti, “A novel related nonce attack for ECDSA,” March 2023. [Online]. Available: https://eprint.iacr.org/2023/305.pdf
7. S. Nakamoto, “Bitcoin: A peer-to-peer electronic cash system,” October 2008. [Online]. Available: http://www.bitcoin.org/bitcoin.pdf
8. V. Buterin, “Ethereum: A next-generation smart contract and decentralized application platform.” December 2014. [Online]. Available: https://ethereum.org/content/whitepaper/whitepaper-pdf/Ethereum_Whitepaper_-_Buterin_2014.pdf
9. “Litecoin github repository.” [Online]. Available: https://github.com/litecoin-project/litecoin
10. “Dogecoin github repository.” [Online]. Available: https://github.com/dogecoin/dogecoin
11. “Dash github repository.” [Online]. Available: https://github.com/dashpay/dash
12. “Bitcoincash github repository.” [Online]. Available: https://gitlab.com/bitcoin-cash-node/bitcoin-cash-node
13. V. Buterin, “Ethereum: A secure decentralised generalised transaction ledger,” April 2014. [Online]. Available: https://ethereum.github.io/yellowpaper/paper.pdf
14. “Bitcoin github repository.” [Online]. Available: https://github.com/bitcoin/bitcoin
15. Standards for Efficient Cryptography Group, “Sec 1: Elliptic curve cryptography,” May 2009. [Online]. Available: https://www.secg.org/sec1-v2.pdf
16. V. Jacquot and B. Donnet, “Oops!. . . i did it again. i reused my nonce.” December 2025. [Online]. Available: https://eprint.iacr.org/TODO
17. Bitcoin Community, “Script,” [Last Accessed: October 30th, 2024]. [Online]. Available: https://en.bitcoin.it/wiki/Script
18. G. Andresen, “Pay to script hash,” Bitcoin, BIP 16, January 2012. [Online]. Available: https://github.com/bitcoin/bips/blob/master/bip-0016.mediawiki
19. E. Lombrozo, J. Lau, and P. Wuille, “Segregated witness (consensus layer),” Bitcoin, BIP 141, December 2015. [Online]. Available: https://github.com/bitcoin/bips/blob/master/bip-0141.mediawiki
20. Bitcoin Community, “Op checksig,” [Last Accessed: October 30th, 2024]. [Online]. Available: https://en.bitcoin.it/wiki/OP_CHECKSIG
21. J. Lau and P. Wuille, “Transaction signature verification for version 0 witness program,” Bitcoin, BIP 143, December 2015. [Online]. Available: https://github.com/bitcoin/bips/blob/master/bip-0143.mediawiki

22. “python-bitcoinlib.” [Online]. Available: https://pypi.org/project/python-bitcoinlib/
23. Sigma Prime, “Solutions for a secure & decentralized world,” [Last Accessed: April 3rd, 2025]. [Online]. Available: https://lighthouse-blog.sigmaprime.io
24. Erigon, “Beyond software frontiers. empowering industries with trustless, decentralized software innovations,” [Last Accessed: April 3rd, 2025]. [Online]. Available: https://erigon.tech
25. @MaryNfs, “Json-rpc api,” [Last Accessed: October 30th, 2024]. [Online]. Available: https://ethereum.org/en/developers/docs/apis/json-rpc/#eth_gettransactionbyhash
26. “eth-account.” [Online]. Available: https://github.com/ethereum/eth-account/tree/main
27. D. Brown, “SEC 2: Recommended elliptic curve domain parameters,” Certicom Research, Standards for Efficient Cryptography 2.0, January 2010.
28. R. Miller, “A glimpse of the deep: Finding a creature in ethereum’s dark forest,” December 2021, [Last Accessed: April 29th, 2025]. [Online]. Available: https://bertcmiller.com/glimpse.html
29. “Cve-2013-7372 detail,” [Last Accessed: May 20, 2025]. [Online]. Available: https://nvd.nist.gov/vuln/detail/cve-2013-7372
30. D. Gilson, “Blockchain.info issues refunds to bitcoin theft victims,” [Last Accessed: May 20, 2025]. [Online]. Available: https://www.coindesk.com/markets/2013/08/21/blockchaininfo-issues-refunds-to-bitcoin-theft-victims
31. T. Pornin, “Deterministic usage of the digital signature algorithm (DSA) and elliptic curve digital signature algorithm (ECDSA),” Internet Engineering Task Force, RFC 6979, August 2013.