POLITECNICO DI TORINO
Repository ISTITUZIONALE

Special Subsets of Addresses for Blockchains Using the secp256k1 Curve

Original
Special Subsets of Addresses for Blockchains Using the secp256k1 Curve / Gangemi, Andrea; Di Scala, Antonio J; Romeo, Giuliano; Vernetti, Gabriele. - In: MATHEMATICS. - ISSN 2227-7390. - 10:15(2022). [10.3390/math10152746]

Availability:
This version is available at: 11583/2970443 since: 2022-08-04T07:41:22Z

Publisher:
MDPI

Published
DOI:10.3390/math10152746

Terms of use:

This article is made available under terms and conditions as specified in the corresponding bibliographic description in the repository

Publisher copyright

(Article begins on next page)

21 November 2024

mathematics

MDPI

Article

Special Subsets of Addresses for Blockchains Using the *secp256k1* Curve

Antonio J. Di Scala [1,†,‡], Andrea Gangemi [1,†,‡], Giuliano Romeo [1,*,†,‡] and Gabriele Vernetti [2,†,§]

[1] DISMA, Department of Mathematical Sciences, Politecnico of Turin, Corso Duca degli Abruzzi, 24, 10129 Torino, Italy; antonio.discala@polito.it (A.J.D.S.); andrea.gangemi@polito.it (A.G.)
[2] DAUIN, Department of Control and Computer Engineering, Politecnico of Turin, Corso Duca degli Abruzzi, 24, 10129 Torino, Italy; gabriele.vernetti@studenti.polito.it
* Correspondence: giuliano.romeo@polito.it
† These authors contributed equally to this work.
‡ Antonio J. Di Scala, Andrea Gangemi and Giuliano Romeo are members of CrypTO, GNSAGA of INdAM and of DISMA Dipartimento di Eccellenza MIUR 2018–2022.
§ Gabriele Vernetti is member of CrypTO.

Abstract: In 2020, Sala, Sogiorno and Taufer were able to find the private keys of some Bitcoin addresses, thus being able to spend the cryptocurrency linked to them. This was unexpected since the recovery of non-trivial private keys for blockchain addresses is deemed to be an infeasible problem. In this paper, we widen this analysis by mounting a similar attack on other small subsets of the set of private keys. We then apply it to other blockchains as well, examining Ethereum, Dogecoin, Litecoin, Dash, Zcash and Bitcoin Cash. In addition to the results, we also explain the techniques we have used to perform this exhaustive search for all the addresses that have ever appeared in these blockchains, and we give an estimate of the time needed to perform all the computations. Finally, we also examine the possibility of mounting a similar attack on other elliptic curves used in blockchains, i.e., Curve25519 and NIST P-256.

Keywords: cryptography; blockchain; Bitcoin; elliptic curves; subsets analysis; addresses; wallet

MSC: 11T71; 68M14; 14H52

check for updates

Citation: Di Scala, A.J.; Gangemi, A.; Romeo, G.; Vernetti, G. Special Subsets of Addresses for Blockchains Using the *secp256k1* Curve. *Mathematics* **2022**, *10*, 2746. https://doi.org/10.3390/math10152746

Academic Editor: Angel Martín-del-Rey

Received: 5 July 2022
Accepted: 29 July 2022
Published: 3 August 2022

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Copyright: © 2022 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

The impact that Bitcoin has had on modern society hardly needs to be explained. In just a few years since the publication of Satoshi Nakamoto's white paper [1], blockchain technology has taken hold all over the world. Its main objective is the creation of an open, public, decentralised and immutable ledger whose reliability is not based on a trusted third party. It is natural that a tool with these features could find use in several other fields besides transaction validation. Some references on the various fields where blockchain technology has been employed can be found in [2].

The blockchain that has caught most of the interest after Bitcoin is perhaps Ethereum, which was theorised in 2013 by Vitalik Buterin [3]. The aim of Ethereum is to be a versatile world programmable computer in which smart contracts—i.e., decentralised programs written in a Turing-complete programming language—can be executed. A mathematical description of Ethereum's blockchain has been published by Gavin Wood in the yellow paper [4].

In a blockchain, the security of transactions is guaranteed by public-key cryptography and, in particular, by the difficulty of solving the discrete logarithm problem over suitable elliptic curves. Using the best-known algorithms, it is practically impossible, in a general case, to recover the private key starting from the knowledge of the public address. For all these reasons, it was quite unexpected when, in 2020, Sala, Sogiorno and Taufer [5] found

the private keys of some existing Bitcoin addresses, being able to spend cryptocurrencies on their behalf. They mounted an attack on a small multiplicative subgroup of a group that is mapped to the group of points of Bitcoin's elliptic curve *secp256k1*. For this small set, a brute-force attack was feasible and, against any odds, four of those addresses coincided with some actually used in Bitcoin's history. The same authors left open the problem of understanding the reasons for such pathological behavior and the analysis of other cryptocurriencies together with other small algebraic structures. Brute-force Bitcoin address generation has also been the object of study in [6].

In this paper, we widen the range of the inspection to seven other subsets of the same order, obtained as cosets of the subgroup used in [5]. Furthermore, we perform the same examination for Ethereum addresses and for some other famous cryptocurrencies that share the same elliptic curve: Dogecoin [7], Litecoin [8], Zcash [9], Dash [10] and Bitcoin Cash [11]. We apply this analysis to all the different encodings used so far by these blockchains. We also explain why this peculiarity may have occurred in Bitcoin, and we discuss the possibility of performing a similar analysis on other important elliptic curves for blockchains, e.g., Curve25519 and NIST P-256. Moreover, we describe, in detail, the techniques we have developed to extract data from the full history of these blockchains and to perform all the queries in a feasible time. To the best of our knowledge, no information on how to efficiently perform such an exhaustive analysis is present in the literature or on any website. We have given public access to all the programming scripts at [12]. Finally, we also provide an estimation of the computational cost, in terms of time and space, needed to perform the inspection.

The paper is organised as follows: in Section 2, we recap some basic definitions from algebra, and we introduce the *secp256k1* curve. In Section 3, we define the small subsets we have decided to analyse, and in Section 4, we summarise the address generation algorithms for both compressed and uncompressed public keys. In Section 5, we describe the strategies employed to extract the complete list of addresses in the shortest possible time, while in Section 6, we report the results we have obtained, and we make a comparison with other elliptic curves that we have found particularly interesting. Finally, in Section 7, we provide some information about the computational time and the space required to perform this analysis, making a comparison between the use of two different computers with different computational power.

2. Preliminaries

Let us recall some basic definitions from algebra and, in particular, from group theory and elliptic curves over finite fields.

2.1. Group Theory

Lemma 1. *Let us consider a cyclic group $G = \langle g \rangle$ of order $n \in \mathbb{N}$. Then, for each divisor d of n there exists a unique subgroup H of order d, and one of its generators is $g^{\frac{n}{d}}$.*

Definition 1 (Coset)**.** *Let us consider a subgroup $H \leq G$ of a commutative group G. A coset of H is the set*

$$gH = Hg = \{gh : h \in H\},$$

where $g \in G$.

It is straightforward to verify that $|gH| = |H|$ and that $gH = H$ if and only if $g \in H$.

Theorem 1. *Let $\mathbb{F}$ be a field. Then the multiplicative group*

$$\mathbb{F}^* = \mathbb{F} \backslash \{0\}$$

is a cyclic group.

2.2. Elliptic Curves and the secp256k1

Definition 2 (Elliptic Curve)**.** *Let $\mathbb{F}$ be a field with a characteristic different from two and three. Let $A, B \in \mathbb{F}$ such that $\Delta = 4A^3 + 27B^2 \neq 0$. Then, we define an elliptic curve $E(\mathbb{F})$ as the following subset of the affine plane over $\mathbb{F}$:*

$$E(\mathbb{F}) = \{(x, y) \in \mathbb{F} \times \mathbb{F} : \ y^2 = x^3 + Ax + B\} \cup \{\mathcal{O}\},$$

where $\mathcal{O}$ denotes the point at infinity.

It is possible to give an additive group structure to the set of points of an elliptic curve $E(\mathbb{F})$ with the inner chord-tangent point-addition (see Section 2 of [13] for more details).

Let us now introduce the most relevant elliptic curve for our purposes, which is the one used by all the cryptocurrencies that we are going to consider.

Definition 3 (The *secp256k1* curve)**.** *Let us consider the prime number*

$$p = 2^{256} - 2^{32} - 2^9 - 2^8 - 2^7 - 2^6 - 2^4 - 1.$$

The elliptic curve

$$E(\mathbb{F}_p) : \ Y^2 = X^3 + 7$$

is called secp256k1.

The number of rational points of the curve *secp256k1* over $\mathbb{F}_p$ is the prime number

$$\begin{aligned} q = \ & 115792089237316195423570985008687907852837564279074904382605163414518161494337, \end{aligned}$$

so that the additive group $\mathcal{E} = E(\mathbb{F}_p)$ is isomorphic to $\mathbb{F}_q(+)$. A generator of this group is the point $P = (P_x, P_y)$, where

$$\begin{aligned} P_x = \ & 55066263022277343669578718895168534326250603453777594175500187360389116729240, \\ P_y = \ & 32670510020758816978083085130507043184471273380659243275938904335757337482424. \end{aligned}$$

Given two points P and $Q = kP$ (obtained by summing k times the point P), it is, in general, very hard to recover the integer k. This is known as the Discrete Logarithm Problem over Elliptic Curves (ECDLP).

3. The Small Subsets

As we have seen in Section 2, the group $E(\mathbb{F}_p)$ of the curve *secp256k1* has prime order

$$\begin{aligned} q = \ & 115792089237316195423570985008687907852837564279074904382605163414518161494337, \end{aligned}$$

and it is then isomorphic to the additive group $\mathbb{F}_q(+)$. Hence, the private key can be chosen among the non-zero elements of $\mathbb{F}_q(+)$, which are $q - 1$. The set of all private keys, then, is in bijection with the multiplicative group $\mathbb{F}_q^*(\cdot)$, which also has order $q - 1$ and, by Theorem 1, is a cyclic group. Therefore, by Lemma 1, it has a unique cyclic subgroup of order d for any divisor d of $q - 1$. Its factorisation is

$$q - 1 = h \cdot p_1 \cdot p_2 \cdot p_3,$$

where

$$\begin{aligned} h &= 18051648 = 2^6 \cdot 3 \cdot 149 \cdot 631, \\ p_1 &= 107361793816595537, \\ p_2 &= 17472360753441437 1449, \\ p_3 &= 341948486974166000522343609283189. \end{aligned}$$

In [5], the authors examined the multiplicative subgroup $H \leq \mathbb{F}_q^*$, whose order is

$$|H| = h = 18051648,$$

since it can be fully investigated in a short time. As pointed out in the same paper, it is also worth analysing some other structures of the same order, for example, the cosets of G. Let us consider the generator $g = 7$ of the group $\mathbb{F}_q^*(\cdot)$. In this way, using Lemma 1 and denoting by

$$\begin{aligned} &g_0 = 7^{p_1 p_2 p_3}, && g_1 = 7^{h p_2 p_3}, && g_2 = 7^{h p_1 p_3}, && g_3 = 7^{h p_1 p_2}, \\ &g_4 = 7^{h p_1}, && g_5 = 7^{h p_2}, && g_6 = 7^{h p_3}, && g_7 = 7^{h}, \end{aligned}$$

we have that

$$\begin{aligned} &|\langle g_0 \rangle| = |H| = h, && |\langle g_1 \rangle| = p_1, && |\langle g_2 \rangle| = p_2, \\ &|\langle g_3 \rangle| = p_3, && |\langle g_4 \rangle| = p_2 p_3, && |\langle g_5 \rangle| = p_1 p_3, \\ &|\langle g_4 \rangle| = p_1 p_2, && |\langle g_7 \rangle| = p_1 p_2 p_3. \end{aligned}$$

The cosets that we investigate in this paper are $g_i H$, for $i \in \{0, \ldots, 7\}$. Notice that $g_0 H = H$, which is the same subgroup considered in [5].

Remark 1. *What the authors have done in [5] is to search among the multiplicative subgroups of $\mathbb{F}_q^*(\cdot)$, even though the private key space is $(E(\mathbb{F}_p), +) \cong \mathbb{F}_q(+)$, hence additive. This observation makes the result (four keys detected) even more surprising since it has been considered a subgroup that is far from the actual nature of group $E(\mathbb{F}_p)$. This very curious fact makes us think that the cause is some implementation error that has occurred during the practical design of the wallets.*

4. Address Generation

In this section, we examine how the addresses are generated starting from the choice of the private key.

4.1. Private and Public Key

The private key is an arbitrary 256-bit integer k. Using k, it is possible to compute the point

$$K = kP = (K_x, K_y)$$

over the elliptic curve, from which the public key is derived. We have two different ways to represent the public keys:

(i) $PK_1 = \text{0x04} \,||\, K_x \,||\, K_y$,

(ii) $PK_2 = \begin{cases} \text{0x02} \,||\, K_x & \text{if } K_y \text{ is even,} \\ \text{0x03} \,||\, K_x & \text{if } K_y \text{ is odd,} \end{cases}$

where we have denoted with $||$ the string concatenation. In the first case, the public key —also known as the *uncompressed public key*—consists of 65 bytes (32 bytes for K_x and K_y, plus the byte 0x04), while in the second case, the public key is called the *compressed public key* and consists of 33 bytes. The ways of obtaining the addresses for each cryptocurrency are discussed in the following paragraphs.

4.2. Bitcoin Addresses

There are three manners to generate Bitcoin addresses starting from the point $K = (K_x, K_y)$.

First of all, two hash functions appear in Bitcoin address generation, which are SHA-256 [14] and RIPEMD-160 [15]. Let us compute, for $i \in \{1, 2\}$:

$$W_i = \text{0x00} \,||\, \text{RIPEMD-160}(\text{SHA-256}(PK_i)),$$
$$checksum_i = (\text{SHA-256}(\text{SHA-256}(W_i)))[1 \ldots 4],$$

where $[1 \ldots 4]$ denotes the first four bytes of that string. The first byte 0x00 is a prefix called *version byte*. Then, the first two ways to generate Bitcoin addresses are, for $i \in \{1, 2\}$,

$$\text{Base58}(W_i \,||\, checksum_i),$$

where Base58 [16] is an encoding scheme.

The addresses of the third kind were introduced in 2017, and they are called *segwit* addresses [17]. These addresses are computed starting only from compressed public keys. First of all, it is computed

$$W = \text{RIPEMD-160}(\text{SHA-256}(PK_2)),$$

Then it is encoded using Bech32 [18], and it is concatenated to the prefix *bc1* in order to obtain the final address format

$$bc1 \,||\, \text{Bech32}(W).$$

4.3. Ethereum Addresses

Ethereum addresses are generated starting from the uncompressed public key; that is

$$PK = \text{0x04}||K_x||K_y.$$

However, a different hash function is used: KECCAK-256 [19]. The address is computed as

$$\text{KECCAK-256}(PK)[1 \ldots 20],$$

where $[1 \ldots 20]$ denotes the first 20 bytes of that string.

After this computation, the addresses are encoded following the rules described in the EIP-55 document [20]. In short, the capitalisation of certain alphabetic characters in the address is changed to obtain a checksum that can be used to protect the integrity of the address from typing or reading errors.

4.4. Dogecoin Addresses

Dogecoin address generation is similar to the first two methods described for Bitcoin; thus we can use uncompressed or compressed public keys. It only changes the *version byte* 0x00 of Bitcoin into 0x1E. In this way, the final Base58 encoding gives a D as the first letter for each address. Dogecoin does not generate addresses following the *segwit* standard.

4.5. Litecoin Addresses

Litecoin can generate addresses using all the three methods described for Bitcoin. In the first two cases, the prefix is the *version byte* 0x30 instead of 0x00. In this way, all the addresses start with the letter L. In the segwit case, the Bech32 encoding of the address is concatenated with the prefix $ltc1$; that is

$$ltc1 \,||\, \text{Bech32}(\text{RIPEMD-160}(\text{SHA-256}(PK_2))).$$

4.6. Dash Addresses

The address generation is similar to Bitcoin, but in this case, the *version byte* is changed into 0x4c, so that all the addresses start with the letter *X*. Dash does not generate addresses following the segwit standard.

4.7. Zcash Addresses

The address generation is similar to Bitcoin, but in this case, the *version byte* is changed into [0x1c, 0xb8]. Zcash addresses can either start with the letter *t* if they are transparent, or *z* if they are shielded. Zcash does not generate addresses following the segwit standard.

4.8. Bitcoin Cash Addresses

Bitcoin Cash (BCH) is the result of a Bitcoin hard fork that happened in 2017 when some Bitcoin nodes did not share the segwit soft fork. BCH addresses are encoded two times:

- first, an address with the same encoding of Bitcoin is obtained,
- second, the address is encoded again with an encoding scheme called *CashAddr* [21], which is used by Bitcoin Cash only. This encoding is similar to Bech32.

BCH addresses that are encoded twice start with the string *bitcoincash:q*, or just with the letter *q*. It comes natural that Bitcoin Cash does not generate addresses following the segwit standard.

5. Experimental Environment and Development

In this section, we describe our experimental environment, which led us to the results shown in the following. In order to perform a deep investigation for the existence of the addresses generated, we filled a database with all addresses that had ever appeared on the blockchains of our interest, exploiting the MySQL *Laragon* tool for the search operation [22].

5.1. Blockchain Addresses Extraction

The extraction of all the addresses ever appeared on a blockchain can be performed in different ways, depending on the time availability. Generally, the most common methods are:

- setting up a *full node* containing a complete local copy of the relative blockchain and reading data from it,
- getting data through public Application Programming Interfaces (APIs), available on the web.

In our case, we decided to set up full nodes when it was impossible to retrieve data from any public API. In this regard, we would like to point out that the availability of public APIs, documentation and general support was very weak for any blockchain other than Bitcoin and Ethereum.

5.2. Development

For some blockchains (Dogecoin, Litecoin, Bitcoin Cash and Ethereum), we developed some Python scripts in order to use the public APIs provided by *Tatum* [23] and *Ankr* [24]. Differently, to analyse Dash and Zcash, we had to build our own full nodes to get the block data from them. Finally, the list of Bitcoin addresses was taken from [25], which offers some interesting information about the Bitcoin blockchain. For anyone interested in learning more, our code is publicly accessible at [12]. We conclude this section by reporting in Table 1 the number of addresses we have examined; that is, the totality of addresses that have ever been used in the history of these blockchains.

Table 1. Number of examined addresses.

Blockchain	Addresses
Bitcoin	923,414,052
Ethereum	144,596,346
Dogecoin	69,817,509
Litecoin	134,530,241
Dash	92,456,113
Zcash	6,813,058
Bitcoin Cash	334,965,092

6. Cosets Examination

In this section, we list the results that we have obtained by inspecting the eight cosets g_iH, $i \in \{0, \ldots, 7\}$, of order $h = 18,051,648$ for the seven blockchains we have chosen to investigate. The eight cosets contain a total of $8h \approx 144$ million addresses. We have checked if these addresses ever appeared in the blockchains mentioned above, i.e., if they have ever held any amount of cryptocurrency.

More specifically, we have written some Python code that, starting from the list of private keys, computes the resulting addresses in the right format and then checks if these addresses ever appeared on the analysed blockchains. The code used in this analysis is publicly accessible at [12]. In Table 2 we sum up all the results.

Table 2. Results of the subsets examination.

Addresses	H	g_iH, $i = 1, \ldots, 7$
Uncompressed Bitcoin	4 addresses	No address
Compressed Bitcoin	3 addresses	No address
Segwit Bitcoin	1 addresses	No address
Ethereum	1 addresses	No address
Dogecoin	3 addresses	No address
Litecoin	2 addresses	No address
Dash	2 addresses	No address
Zcash	No address	No address
Bitcoin Cash	6 addresses	No address

From this analysis, it turns out that only three addresses are non-trivial, i.e., they do not have 1 or -1 as the private key. Notice that, for example, in Dogecoin, we have found three trivial addresses since each private key can be associated with two different encodings of the public key (compressed and uncompressed), for a total of four potential trivial addresses. The three non-trivial addresses belong to the Bitcoin blockchain. The two addresses already found by Sala et al. [5] were generated in 2013 and 2014, so they are present in Bitcoin Cash as well, while the third one is the address

1H1jFxaHFUNT9TrLzeJVhXPyiSLq6UecUy,

and it was generated starting from a compressed public key. It was created on 15 October 2019, after the Bitcoin Cash fork, which is dated 1 August 2017. The address has no cryptocurrency in it nowadays.

The story of the address

1PSRcasBNEwPC2TWUB68wvQZHwXy4yqPQ3,

already found by Sala et al., is somewhat interesting. That address was created on 15 March 2014, and the funds got removed in June 2018 only after the authors of the previously cited papers contacted them (read [5] to know more). Hence, the address is old enough to appear on BCH. It is indeed present:

qrmzrdndlfxpnkk3w5d5l7etnysnqfgk5yxsf6k0qq,

after the new encoding with the CashAddress format.

However, the address is empty as someone moved the funds on that address on 1 May 2019. Furthermore, on 15 November 2018, Bitcoin Cash had yet another hard fork that led to the birth of *Bitcoin SV*. Hence, this address must be present on that blockchain as well. By examining a Bitcoin SV explorer, it turns out that the funds from that address also got moved on 1 May 2019, two minutes earlier than the transaction on Bitcoin Cash. It seems likely to assume that the funds got moved from the same entity.

Finally, notice that we were not able to find any addresses on the seven cosets we have chosen to examine. This is interesting but not unexpected, as the probability of finding an address with this brute-force method is really low, as already underlined in [5]. In this way, the security of these blockchains is not threatened since it shows that it is way more profitable to honestly mine the protocol than trying to generate random private keys with the aim of stealing cryptocurrency [6]. This may also confirm that the non-trivial addresses were generated due to poor implementation of some wallets. It would also be interesting to try to understand which Bitcoin wallets generated these addresses.

6.1. Other Curves to Examine

The most used elliptic curve in blockchains, besides the *secp256k1*, is for sure the *Curve25519*. It is employed in several blockchains, including Monero [26], Cardano [27], Solana [28] and Algorand [29]. The defining curve is the Montgomery curve

$$E(\mathbb{F}_p): \; Y^2 = X^3 + 486662X^2 + X,$$

where $p = 2^{255} - 19$ is a prime number. The number of rational points is $n = 8l$ where

$$l = 2^{252} + 27742317777372353535851937790883648493.$$

Reasoning, as in the case of the curve *secp256k1*, the private key can be chosen among the non-zero points, whose number is

$$l - 1 = 2^2 \cdot 3 \cdot 11 \cdot q_1 \cdot q_2,$$

where

$$\begin{aligned} q_1 &= 27660262428164223993721868055713982666874 7, \\ q_2 &= 19821142323093075401308452576369 7. \end{aligned}$$

The private key space is then in bijection with $\mathbb{Z}_l^*(\cdot)$, it is cyclic, and it has a unique subgroup for each divisor of its order $l - 1$, by Lemma 1. Notice that an analogue analysis can not be performed since the small subgroup contains only $132 = 2^2 \cdot 3 \cdot 11$ points.

Finally, another curve that might be worth analysing in the future is the NIST *P-256* curve, defined in the specification [30]. It is actually used by NEO [31], Tezos [32] and Ontology [33]. The curve is

$$E(\mathbb{F}_p): \; Y^2 = X^3 - 3X + B,$$

where

$$\begin{aligned} B = \; & 41058363725152142129326129780047268409114441015993725 5 \\ & 5483525631403946740129 1, \end{aligned}$$

and p is the prime number

$$p = 2^{256} - 2^{224} + 2^{192} + 2^{96} - 1.$$

The order of this curve is

$$q = 115792089210356248762697446949407573529996955224135760 \\ 342422259061068512044369,$$

and $q-1$ factorises as

$$q-1 = 2^4 \cdot 3 \cdot 71 \cdot 131 \cdot 373 \cdot 3407 \cdot 17449 \cdot 38189 \cdot 187019741 \cdot 622491383 \cdot q_1 \cdot q_2,$$

where

$$q_1 = 26247475503338692784167739 53,$$
$$q_2 = 1002328039319.$$

This case is certainly more interesting than the previous one, since there are several subgroups of order dividing $q-1$ that can be inspected.

7. Computational Benchmarks

The analysis consists of three steps:

(1) The creation of the *secp256k1_keys.txt* file, in each row containing the information about the private keys and the corresponding x and y coordinates of the public keys of our special subsets. This file contains around 144 million rows, and it weights around 29 GB.
(2) The downloading of the lists of addresses and uploading of the same lists on the MySQL Laragon tool. Information about the total number of addresses has already been given in Section 5.
(3) The generation of blockchain addresses in our subsets starts from the file generated during the first step, searching for some matches with the lists uploaded during the second step.

To perform most of these computations, we used a desktop computer with processor Intel Core i9-10900K and a Non-Volatile Memory with 1 TB of capacity. We then compared the execution time with a laptop computer using a processor Intel Core i7-11370H and 16 GB of RAM.

Step 1 must be executed just once, and then the file can be used every time we need in order to perform a different analysis. We generated this file using the laptop computer, and it required around 25 h.

Step 2 is the most time-consuming operation. We uploaded all the lists on the desktop computer. The time needed depends on the total number of addresses in each list. For example, for Bitcoin, we needed around one week to upload the address list to Laragon, while the same operation for Ethereum took about five days. The size of these lists ranges from 58 GB to store the entire Bitcoin address list to just over 0.5 GB to store the entire Zcash address list. We then repeated the same operations for the laptop computer, and, on average, we needed around 15% more time.

Finally, Step 3 must be performed once per every different generation method, i.e., 15 different scripts must be executed. On average, every script took around 17 h of time on the computer mounting the i9 processor. On the laptop, the required time was around 12% longer.

8. Conclusions

In this paper, we have given an answer to two open questions left in [5]. It turns out that the strange behaviour observed by Sala et al. is a peculiarity of Bitcoin and its hard forks, where we have also found a new address not detected in [5]. Instead, we have not been able to find any non-trivial private keys for the other analysed blockchains and for the cosets of the starting subgroup. Another contribution of this work is the comprehensive

analysis of the addresses we have extracted from some of the most important blockchains to date. In fact, it is almost impossible to find relevant information on the web for any blockchain other than Bitcoin and, to some extent, Ethereum. The interested readers can extract addresses with the same methods using the scripts available in our GitHub folder [12]. For future work, it might be interesting to analyse more thoroughly the wallets used in the period these addresses were generated to try to confirm the hypothesis that this behavior really comes from an incorrect implementation of the wallet itself. In addition, it is possible to examine the small subgroups found in Section 6.1 to understand if the NIST P-256 curve also presents some strange behaviour on those subsets.

Author Contributions: Conceptualisation, A.J.D.S., A.G., G.R. and G.V.; Investigation, A.J.D.S., A.G., G.R. and G.V.; Methodology, A.J.D.S., A.G., G.R. and G.V.; Validation, A.J.D.S., A.G., G.R. and G.V.; Supervision, A.J.D.S., A.G., G.R. and G.V.; Writing—original draft, A.J.D.S., A.G., G.R. and G.V.; Writing—review and editing, A.J.D.S., A.G., G.R. and G.V.; Software A.J.D.S., A.G., G.R. and G.V. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: All the data and the results were obtained using programming scripts that are publicly accessible in our GitHub at https://github.com/GitGab19/blockchain-addresses-list, accessed on 4 July 2022.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Nakamoto, S. Bitcoin: A Peer-to-Peer Electronic Cash System. 2008. Available online: https://bitcoin.org/bitcoin.pdf (accessed on 4 July 2022)
2. Pilkington, M. Blockchain Technology: Principles and Applications. In *Research Handbook on Digital Transformations*; Edward Elgar Publishing: Cheltenham, UK, 2016.
3. Buterin, V. Ethereum: A Next-Generation Smart Contract and Decentralized Application Platform. 2014. Available online: https://ethereum.org/en/whitepaper (accessed on 4 July 2022).
4. Wood, G. Ethereum: A Secure Decentralised Generalised Transaction Ledger. 2014. Available online: https://ethereum.github.io/yellowpaper/paper.pdf (accessed on 4 July 2022).
5. Sala, M.; Sogiorno, D.; Taufer, D. A Small Subgroup Attack on Bitcoin Address Generation. *Mathematics* **2020**, *8*, 1645. [CrossRef]
6. Brighente, A.; Camaioni, M.; Conti, M.; Olivastri E. Should I Mine or Should I Break: On the Worthiness of Brute-Forcing Cryptocurrency Addresses. In Proceedings of the 2022 IEEE International Conference on Pervasive Computing and Communications Workshops and Other Affiliated Events (PerCom Workshops), Pisa, Italy, 21–25 March 2022; pp. 279–284.
7. Dogecoin Whitepaper. Available online: https://coinnws.com/wp-content/uploads/2019/08/dogecoin-whitepaper.pdf (accessed on 4 July 2022).
8. Litecoin Whitepaper. Available online: https://www.allcryptowhitepapers.com/litecoin-whitepaper/ (accessed on 4 July 2022).
9. Bowe, S.; Hopwood, D.; Hornby, T.; Wilcox, N. Zcash Protocol Specification. 2016. Available online: https://zips.z.cash/protocol/protocol.pdf (accessed on 4 July 2022).
10. Dash Whitepaper. Available online: https://github.com/dashpay/docs/raw/master/binary/Dash%20Whitepaper%20-%20V2.pdf (accessed on 4 July 2022).
11. Bitcoin Cash Specifications. Available online: https://bitcoincash.org/ (accessed on 4 July 2022).
12. Our GitHub Folder. Available online: https://github.com/GitGab19/blockchain-addresses-list (accessed on 4 July 2022).
13. Washington, L.C. *Elliptic Curves: Number Theory and Cryptography*, 2nd ed.; Chapman & Hall/CRC: Boca Raton, FL, USA, 2008.
14. Penard, W.; van Werkhoven, T. Chapter 1: On the Secure Hash Algorithm Family. In *Cryptography in Context*; 2007. Available online: https://webspace.science.uu.nl/~tel00101/liter/Books/CrypCont.pdf (accessed on 4 July 2022).
15. Bosselaers, A.; Dobbertin, H.; Preneel, B. RIPEMD-160: A Strengthened Version of RIPEMD. In *International Workshop on Fast Software Encryption*; Springer: Berlin/Heidelberg, Germany, 1996; pp. 71–82.
16. Base58 Encoding. Available online: https://tools.ietf.org/id/draft-msporny-base58-01.html (accessed on 4 July 2022).
17. Segwit Encoding. Available online: https://en.bitcoin.it/wiki/BIP-0173 (accessed on 4 July 2022).
18. Bech32 Encoding. Available online: https://en.bitcoin.it/wiki/Bech32 (accessed on 4 July 2022).
19. Bertoni, G.; Daemen, J.; Peeters, M.; Van Assche, G. Keccak. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*; Springer: Berlin/Heidelberg, Germany, 2013; pp. 313–314.

20. EIP55 Specifications. Available online: https://github.com/Ethereum/EIPs/blob/master/EIPS/eip-55.md (accessed on 4 July 2022).
21. Bitcoin Cash Addresses. Available online: https://github.com/bitcoincashorg/bitcoincash.org/blob/master/spec/cashaddr.md (accessed on 4 July 2022).
22. Laragon. Available online: https://laragon.org/ (accessed on 4 July 2022).
23. Tatum. Available online: https://tatum.io/ (accessed on 4 July 2022).
24. Ankr. Available online: https://www.ankr.com/ (accessed on 4 July 2022).
25. All Bitcoin Addresses. Available online: http://alladdresses.loyce.club/?C=M;O=D (accessed on 4 July 2022).
26. Alonso, K.M.; Noether, S. Zero to Monero. 2020. Available online: https://www.getmonero.org/library/Zero-to-Monero-2-0-0.pdf (accessed on 4 July 2022).
27. Hoskinson, C. Why We Are Building Cardano. 2017. Available online: https://whitepaper.io/document/581/cardano-whitepaper (accessed on 4 July 2022).
28. Yakovenko, A. Solana: A New Architecture for a High Performance Blockchain. 2018. Available online: https://coincode-live.github.io/static/whitepaper/source001/10608577.pdf (accessed on 4 July 2022).
29. Chen, J.; Micali, S. Algorand. 2017. Available online: https://arxiv.org/pdf/1607.01341.pdf (accessed on 4 July 2022).
30. Chen, L.; Moody, D.; Regenscheid, A.; Randall, K. *Recommendations for Discrete Logarithm-Based Cryptography: Elliptic Curve Domain Parameters*; Draft NIST Special Publication 800-186; 2019. Available online: https://csrc.nist.gov/publications/detail/sp/800-186/draft (accessed on 4 July 2022)
31. Neo Specifications. Available online: https://docs.neo.org/docs/en-us/index.html (accessed on 4 July 2022).
32. Goodman, L.M. Tezos—A Self-Amending Crypto-Ledger. 2014. Available online: https://tezos.com/whitepaper.pdf (accessed on 4 July 2022).
33. Ontology Specifications. Available online: https://ont.io/wp/Ontology-Introductory-White-Paper-EN.pdf (accessed on 4 July 2022).