Research Paper

April 2019

Ethercombing:

Finding Secrets in Popular Places

Adrian Bednarek

Ethercombing: Finding Secrets in Popular Places

Adrian Bednarek <abednarek@securityevaluators.com>

Independent Security Evaluators, LLC
April 2019

Abstract:

Blockchains are public ledgers of transactions verified through the use of public and private keys to sign and prove ownership of transaction data. Popular blockchains have hundreds of millions of transactions which include some of the most popular -- Bitcoin, Waves, Ripple, ZCash, Monero and Ethereum. Currently, on the Ethereum blockchain there are 345 million transactions [1] across 47 million [2] key pairs. The chance of generating a private key already used on the blockchain is around 1 in 2^{256} – all but impossible.

In this paper we examine how, even when faced with this statistical improbability, ISE discovered 732 private keys as well as their corresponding public keys that committed 49,060 transactions to the Ethereum blockchain. Additionally, we identified 13,319 Ethereum that was transferred to either invalid destination addresses, or wallets derived from weak keys that at the height of the Ethereum market had a combined total value of $18,899,969. In the process, we discovered that funds from these weak-key addresses are being pilfered and sent to a destination address belonging to an individual or group that is running active campaigns to compromise/gather private keys and obtain these funds. On January 13, 2018, this "blockchainbandit" held a balance of 37,926 ETH valued at $54,343,407.

Keywords: Blockchain, Public key, Private key Encryption, Ethereum, Cryptography, Brute force, Entropy, Key truncation

Introduction:

This paper focuses on our discovery of private keys used to commit Ethereum blockchain transactions. The probability of encountering a private key that corresponds to someone else's Ethereum address is around 1 in 2^{256}. To cover just 1% of that key space, even if we used computing resources that would allow us to generate 100 trillion keys per second, it would take us roughly 3.6×10^{53} years. However, instead of attempting to brute force search random private keys, we devised ways to discover keys that may have been generated using faulty code, faulty random number generators, or a combination of both. The following sections outline how an Ethereum address is generated and our approach to discover those private keys that were generated in suboptimal ways.

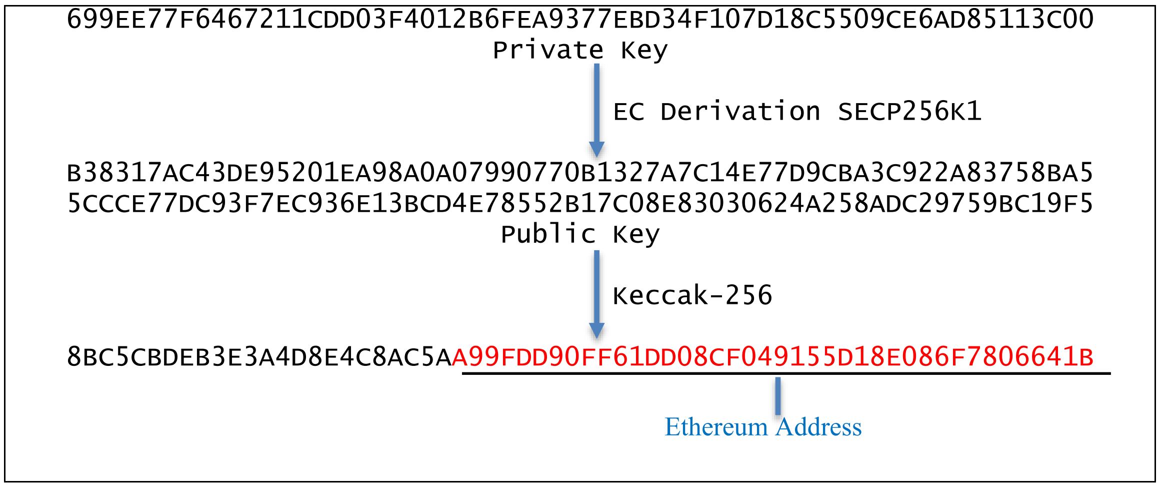
Background:

The Ethereum project uses elliptic curve cryptography to generate the public/private key pair. The 256-bit private key is used to compute a point on the secp256k1 ECDSA curve to generate the public key. The public key is then hashed using keccak256. That hash is truncated to the lower 160 bits to produce the public Ethereum address. The Ethereum address cannot be reversed back into a public key, nor can the Ethereum address be used in any way to derive the underlying private key that was used to generate it.

Given a randomly generated private key, p_r, that is within the valid range of one to the maximum value defined by the secp256k1 curve, 0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFEBAAEDCE6AF48A03BBFD25E8CD0364140, we can calculate the public Ethereum address by taking the lower (right most) 160 bits of the keccack256 hash of the public key, as defined by:

$$A(p_r) = B_{0..159}(KECCAK256(ECDSAPUBKEY(p_r)))$$

Figure 1 illustrates the workflow to derive an Ethereum address from a randomly generated 256-bit private key.



```
699EE77F6467211CDD03F4012B6FEA9377EBD34F107D18C5509CE6AD85113C00
B38317AC43DE95201EA98A0A07990770B1327A7C14E77D9CBA3C922A83758BA5
5CCCE77DC93F7EC936E13BCD4E78552B17C08E83030624A258ADC29759BC19F5
8BC5CBDEB3E3A4D8E4C8AC5AA99FDD90FF61DD08CF049155D18E086F7806641B
```

Figure 1.Example flow of deriving an Ethereum address from a private key.

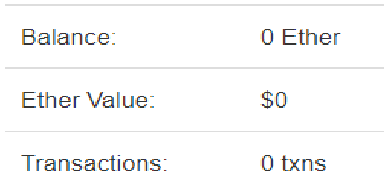
Knowing this algorithm, the goal of our research was to find Ethereum addresses that could not have plausibly been generated by a correct implementation of the algorithm, or, that were correctly derived from non-random private keys.

Experiment Design:

The Ethereum blockchain allows anyone to query an address for information[1], such as balances, transfers, and committed transactions. This is done by querying an Ethereum node which can be run locally or remotely. Or for ease of use, several online services encapsulate the underlying data via web interfaces. One such tool, Etherscan [1], can be used to query the public Ethereum address from the above example:

- A99FDD90FF61DD08CF049155D18E086F7806641B

One can navigate to https://etherscan.io/address/A99FDD90FF61DD08CF049155D18E086F7806641B and see that, not surprisingly, there are 0 transactions for this address[2].



Balance:	0 Ether
Ether Value:	$0
Transactions:	0 txns

Figure 2. Etherscan.io lookup of A99FDD90FF61DD08CF049155D18E086F7806641B.

With nearly 50 million public Ethereum addresses having recorded transactions on the Ethereum blockchain, it is likely that we may encounter keys that are weak or lack randomness, due to several possible factors. An obvious one is key truncation. That is, where a random 256-bit private key is generated but only a small subset is used due to coding/complier/framework or other unknown errors.

For example, a 256-bit private key with the value of:

- `0x47579DA2BEA463533DBFAD6FCF8E90876C2FE9760DC1162ACC4059EE37BDDB5C`

If truncated to 32 bits, would result in the following key:

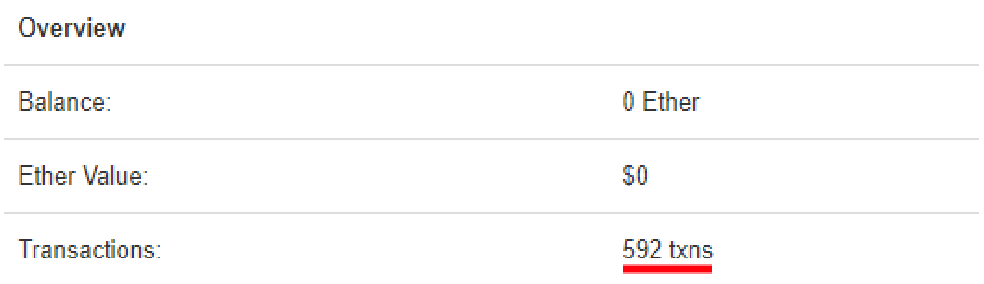
- `0x000000000000000000000000000000000000000000000000000000037BDDB5C`

[1] https://web3js.readthedocs.io/en/1.0/

[2] At the time of this writing, there were no transactions at this address. However, as the key is now disclosed in this paper, the address is public and may have since then been used for transactions. This key and Ethereum address should no longer be used for performing Ethereum transactions!

In an experiment, we picked a private key of 1, for no reason other than that it is the lower bound of a possible private key for secp256k1 and it also lies within the 1 to 2^{32}-1 range of a 32-bit truncated key. We use the private key 0x0000000000000000000000000000000000000000000000000000000000000001 to derive the public Ethereum address 0x7e5f4552091a69125d5dfcb7b8c2659029395bdf.

As previously discussed, recall the infinitesimal probability of two Ethereum users generating the same private key—assuming at least one user is generating them randomly. It should not matter whether we explore the key space from the lower bound, upper bound, keys using the digits of pi, randomly, or so on—there should be no coincidences with another Ethereum user's randomly generated key. Instead, using Etherscan.io to query transaction data on the above public address derived from a private key of 0x01 we are presented with the following evidence of a collision with our intentionally generated key (i.e., the key is or was in use):



Overview

Balance:	0 Ether
Ether Value:	$0
Transactions:	592 txns

Figure 3. Etherscan.io lookup of 7e5f4552091a69125d5dfcb7b8c2659029395bdf.

ISE revealed that there are 592 transactions on an Ethereum address derived from a private key of 0x01, and no Ether currently stored at that address. If a private key is chosen at random then the chances of someone else generating that same key are approximately 1 in 2^{256}, which is for all practical purposes a 0% chance. Since a private key of 0x01 has approximately zero percent chance of occurring randomly, we must assume this value was either chosen on purpose or due to an error. The following sections detail our search to understand and examine how widespread the generation of weak keys is in the Ethereum blockchain.

Scope of Research:

Our research sought to locate Ethereum addresses based on the use of weak keys, and to examine how those addresses are used. While it is improbable that a weak key would ever be generated under legitimate circumstances using the appropriate code paths, we hypothesized that weak private keys may still be generated by coding mistakes, or operating system, device, and execution environment errors, and that these issues are common. Aside from key truncation, some other common mistakes that could weaken 256-bit keys are:

- Code logic errors
- Type confusion
- Entropy errors
- Random device errors
- Poorly handled exceptions
- Memory reference errors
- Memory corruption
- Seed Re-use
- Malware compromise

Due to limited computing resources, it is not feasible to enumerate all keys even in a much smaller 64-bit key space. So, instead we focus on the achievable: enumerating keys that would appear in a smaller 32-bit subset of the 256-bit private key. This amounts to 4,294,967,295 private keys for which we will need to calculate the corresponding public Ethereum address for and query the blockchain.

Methodology:

To perform bulk scanning of potential Ethereum addresses, it is impractical, and even abusive in terms of resource usage, to query an online service like Etherscan. Instead, we generated an in-memory hash map of all public Ethereum addresses and queried this in-memory data structure for each enumerated key. On a local mid-range laptop this resulted in a performance of roughly 15,000 key generations and lookups per second, per CPU core, with the bottleneck being the ECDSA private to public key generation portion.

We focus on eight 32-bit "sub-regions" in the 256-bit key space where we are likely to observe Ethereum addresses in use that have resulted from a weak private key. We expected that the lower 32-bit portion of the key space would be most likely to contain weak keys; to account for endianness, we also scanned the upper 32-bit portion, and for thoroughness we tested each distinct section of the 256-bit key space with a 32-bit window which may yield keys. To illustrate the regions we scanned, Figure 4, below, depicts each region we have identified for enumeration. While enumerating the 32-bit key space of each region (A through H) we leave the remaining 224 bits of the 256-bit key set to 0x00.

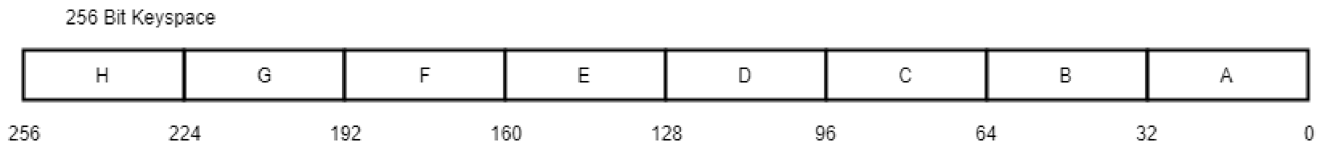


Figure 4. 256-bit key space represented by parts H through A.

This gives us eight regions with a possible 2^{32} -1 (i.e., ~4.3 billion) combinations per region. Translating the region definitions into explicit private key ranges, we scanned and tested these key ranges for transaction activity on the Ethereum blockchain:

```
Group A
0000000000000000000000000000000000000000000000000000000000000001 to
00000000000000000000000000000000000000000000000000000000FFFFFFFF

Group B
0000000000000000000000000000000000000000000000000000000100000000 to
000000000000000000000000000000000000000000000000FFFFFFFF00000000

Group C
0000000000000000000000000000000000000000000000010000000000000000 to
0000000000000000000000000000000000000000FFFFFFFF0000000000000000

Group D
0000000000000000000000000000000000000001000000000000000000000000 to
00000000000000000000000000000000FFFFFFFF000000000000000000000000

Group E
0000000000000000000000000000000100000000000000000000000000000000 to
000000000000000000000000FFFFFFFF00000000000000000000000000000000

Group F
0000000000000000000000010000000000000000000000000000000000000000 to
0000000000000000FFFFFFFF0000000000000000000000000000000000000000

Group G
0000000000000001000000000000000000000000000000000000000000000000 to
00000000FFFFFFFF000000000000000000000000000000000000000000000000

Group H
0000000100000000000000000000000000000000000000000000000000000000 to
FFFFFFFF00000000000000000000000000000000000000000000000000000000
```

The above key space ranges, while making up an infinitesimal part of the 256-bit key space, are some areas that private keys might exist in due to errors or other factors compromising randomness of a 256-bit key. The following section outlines our results for each of the eight key space ranges.

Results:

We discovered 49,060 transactions spread over 732 public keys for which we have the private key, with a total transfer amount of more than 32 Ethereum. The present-day balance across these keys was 0 Ethereum, however that balance is volatile since there are daily transfers in and out of those addresses.

Present day balances for other types of cryptocurrency across these 732 public keys amounted to 60,286,012 ERC-20[3] based tokens. Tokens exist on Ethereum via smart contracts. However, enumerating all ERC-20 based transactions and the sum of those transactions is beyond scope of this research and may be included in follow up future work.

The figure below, Figure 5, is a graphical depiction of the 732 private keys we discovered that were used for blockchain transactions. The y-axis is the value of a key at its corresponding group offset, and the keys are plotted from left to right in order of increasing Ethereum address. For example, in group A, the bulk of discovered keys existed below the 255 or 0x000000FF boundary.

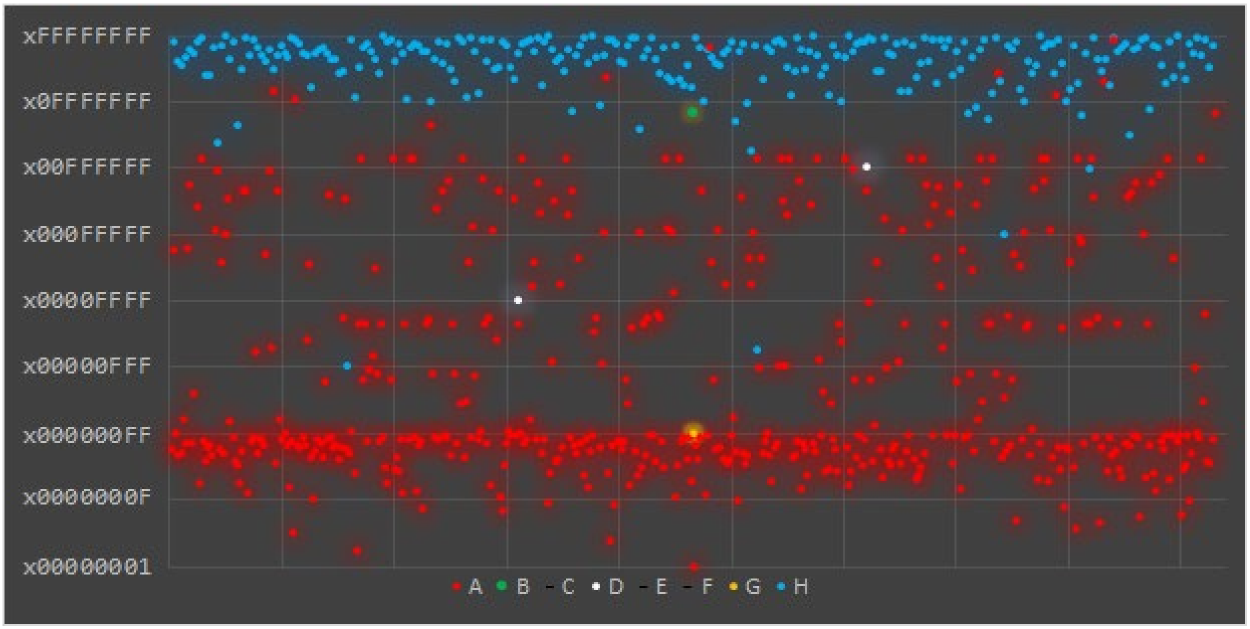


Figure 5. Tally of all private/public keypairs we have access to from Groups A through H.

For each group we list the key range scanned, number of keys found, number of total transactions, total amount for those transactions, and the combined present balance of the addresses. Groups B, C, D, E, F, G are combined into one section since there were only 4 keys found amongst those regions.

Group A:

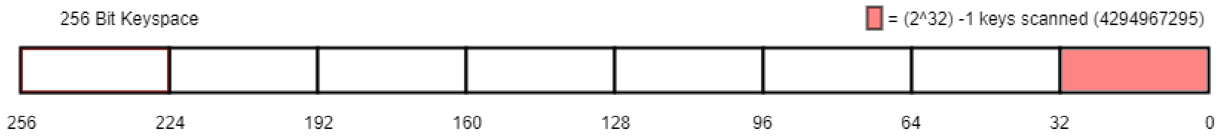


Figure 6. Group A.

[3] https://eips.ethereum.org/EIPS/eip-20

Group A spans the key range of:

```
0000000000000000000000000000000000000000000000000000000000000001 to
00000000000000000000000000000000000000000000000000000000FFFFFFFF
```

Scanning this region of the key space yielded 8,920 transactions through 464 private keys. The total value of transactions using these weak private keys was 28.9456 Ethereum. While transactions are common in this range, there is currently a balance of 0 ETH. Figure 7, below depicts private keys and their offset of this group.

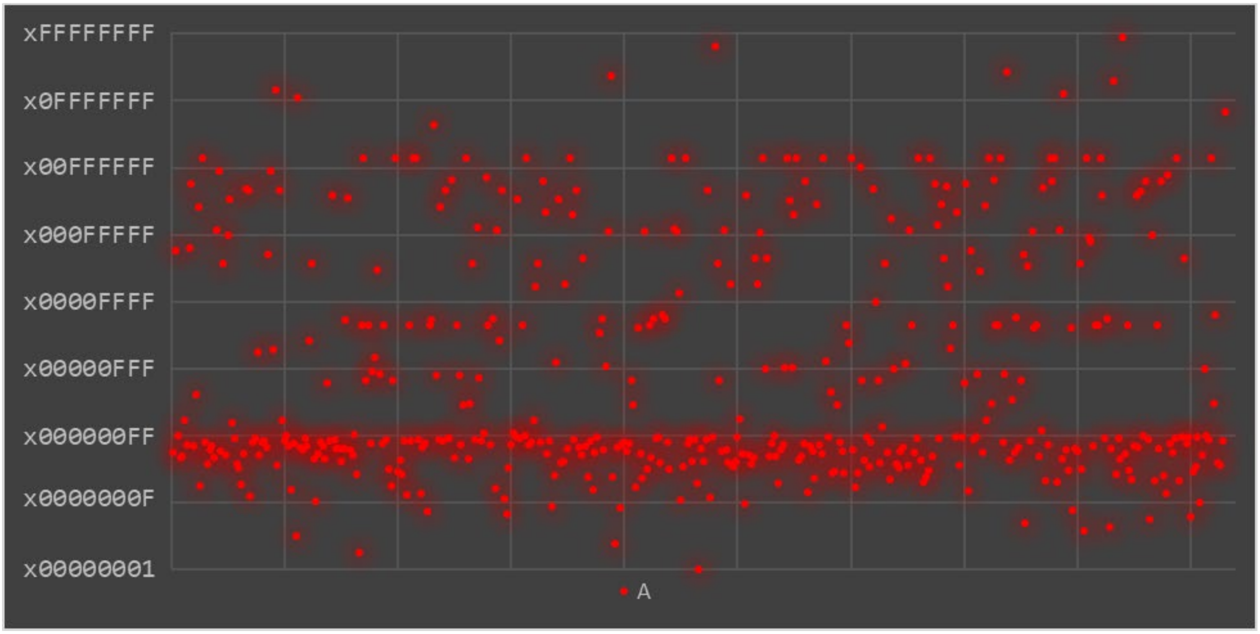


Figure 7. Group A. 8920 transactions spread amongst 464 private keys totaling 28.9456 in Ethereum.

Groups B, C, D, E, F, G:

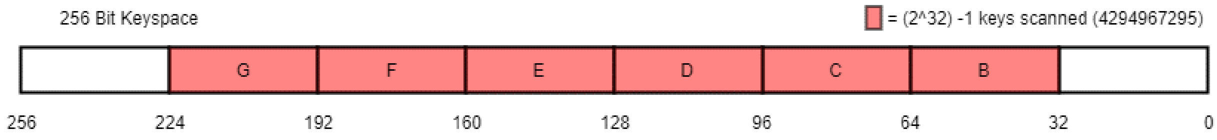


Figure 8. Groups B, C, D, E, F, G.

Groups B, C, D, E, F, G span the ranges of:

```
0000000000000000000000000000000000000000000000000000000100000000 to
000000000000000000000000000000000000000000000000FFFFFFFF00000000

0000000000000000000000000000000000000000000000010000000000000000 to
0000000000000000000000000000000000000000FFFFFFFF0000000000000000

0000000000000000000000000000000000000001000000000000000000000000 to
00000000000000000000000000000000FFFFFFFF000000000000000000000000

0000000000000000000000000000000100000000000000000000000000000000 to
000000000000000000000000FFFFFFFF00000000000000000000000000000000

0000000000000000000000010000000000000000000000000000000000000000 to
0000000000000000FFFFFFFF0000000000000000000000000000000000000000

0000000000000001000000000000000000000000000000000000000000000000 to
00000000FFFFFFFF000000000000000000000000000000000000000000000000
```

Groups B through G are merged into a single section since they only yielded 4 private keys that committed 29 transactions on the blockchain. The total amount of ETH transferred was <0. This result was not surprising as we expected to find a majority of our findings to be in either end of the 256-bit key space via our narrow 32-bit window. Interestingly, one key in these groups, key value of 0x0a000000 in group B had an outgoing transfer to an address that currently holds 44,744 ETH, more on that later. Figure 9 illustrates these 4 keys and their respective offset of their color-coded group.

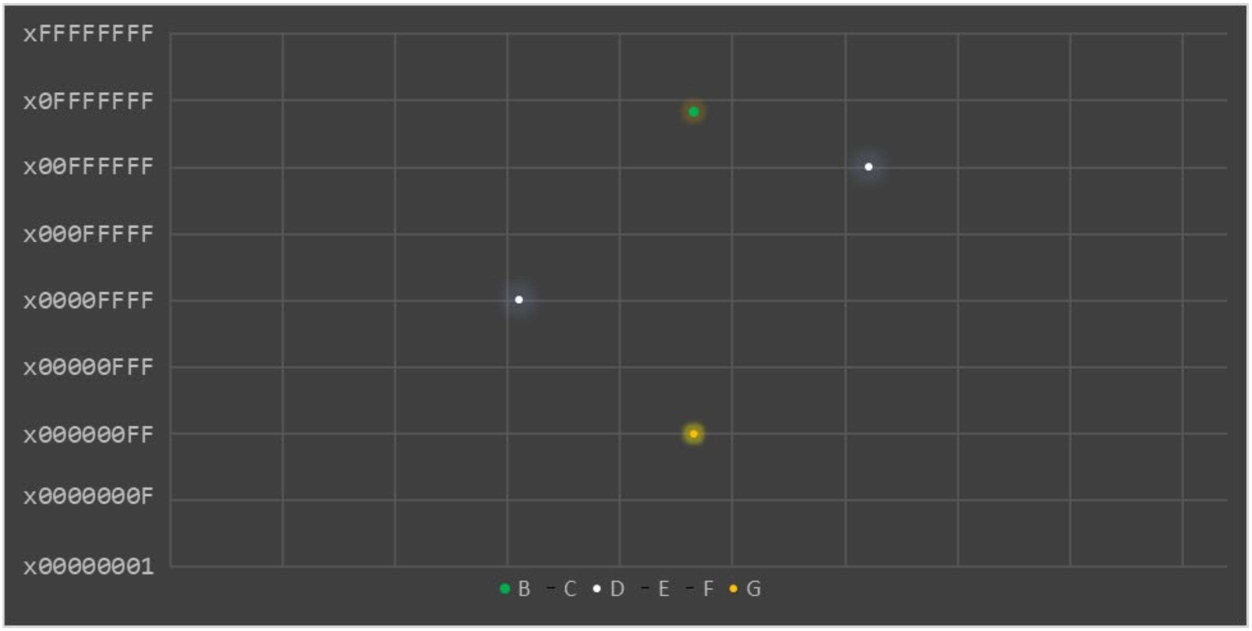


Figure 9. Groups B through G. 29 transactions via 4 keys.

Group H:

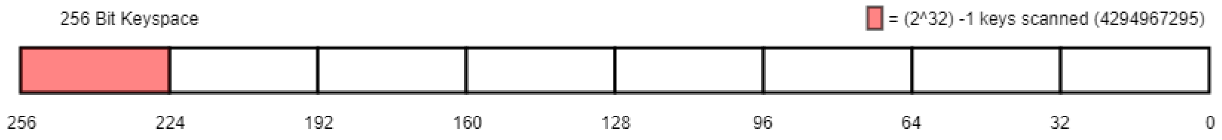


Figure 10. Group H.

Group H spans the key range of:

```
0000000100000000000000000000000000000000000000000000000000000000 to
FFFFFFFF00000000000000000000000000000000000000000000000000000000
```

Scanning this region of the key space yielded 40,111 transactions through 264 private keys. The total value of transactions using weak private keys was 3.38 Ethereum. Like group A, transactions are common in this range so the balance in this group fluctuates, however there is currently a balance of 0 ETH. The figure below, Figure 11, depicts private keys and their offset (y-axis) of this group.

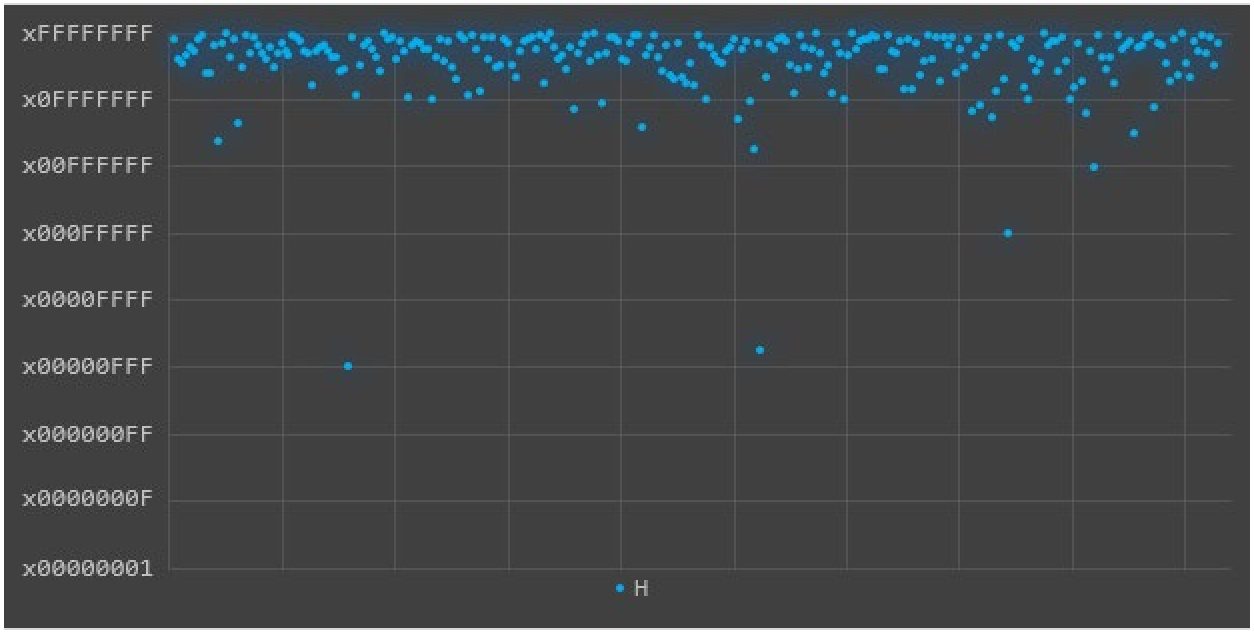


Figure 11. Group H. 40,111 transactions over 264 private keys totaling 3.380 Ethereum.

Analysis and Observations:

Fortunately, using weak private keys is not a widespread problem. However, it was surprising to encounter 732 private keys that we could access. It is also surprising to see a combined transaction volume of 49,060. Apparently, there are transactions committed to these weak keys every day, and potentially new weak keys being introduced as time goes on. The total computing time to scan 8 regions of the 256-bit key space via 32-bit windows, or regions, was ~128 CPU hours per region or 1024 hours total. Luckily, this task can be split up and run in parallel and we were able divide and scan all 8 32-bit regions which resulted in generating and checking ~34.3 billion keys in an 8-hour period. So, it is feasible to run periodic checks to identify new private keys that have been introduce in future works.

"Blockchainbandit"

Previously, we noted an address in group B at offset 0x04000000 that was interacted with via an outbound transaction to a destination address of 0x957cd4ff9b3894fc78b5134a8dc72b032ffbc464, an individual we refer to as the 'blockchainbandit'. After a cursory examination of this transaction, we discovered that the destination address belongs to an individual or group that is running active campaigns to compromise/gather private keys and take these funds. Since the price of Ethereum varies wildly over time, this group or individual has held at most, $54,343,407 worth of ETH, and currently, due to a major market correction, $6.1 million through a present balance of 44,744 ETH.

Noting that the 'blockchainbandit' has amassed millions worth of Ethereum through ~5,400 transactions, we wanted to see how quickly funds would be looted if we populated an Ethereum address we hold common knowledge of the private key to.

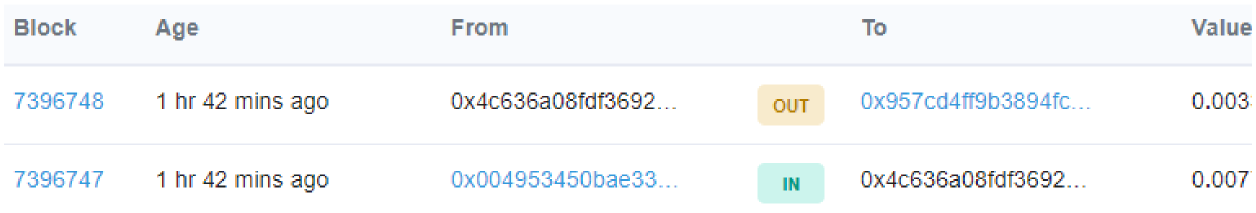
Both ISE and the 'blockchainbandit' have knowledge of the following private key:

- 0x00000000000000000000000000000000000000000000000000a00000000000000

Which is represented by the following derived Ethereum address:

- 0x4c636a08fdf3692a9bca111e8a7f4a0e28eb4457

ISE sent ~1$ worth of Ethereum to 0x4c636a08fdf3692a9bca111e8a7f4a0e28eb4457, an address that was last interacted with over 285 days ago on June 7, 2018. The 'blockchain' bandit instantly sent an outbound transaction from this address to their holding address of 0x957cd4ff9b3894fc78b5134a8dc72b032ffbc464. Figure 12, is an excerpt from Etherscan.io detailing these series of events[4].



Block	Age	From		To	Value
7396748	1 hr 42 mins ago	0x4c636a08fdf3692…	OUT	0x957cd4ff9b3894fc…	0.003
7396747	1 hr 42 mins ago	0x004953450bae33…	IN	0x4c636a08fdf3692…	0.007

Figure 12. 'Blockchainbandit' loots ~1$ worth of ETH immidetly after ISE deposits it to a weak private key.

While performing additional research to get a grasp on why or how these weak keys are being generated, we also encountered additional, more devastating issues dealing with passphrase wallets, faulty address generation, and faulty destination address targets that are outlined in the sections below.

Parity Wallet with Empty Passphrase

The Ethereum address 0x00a329c0648769a73afac7f9381e08fb43dbea72 is derived from a private key of 0x4d5db4107d237df6a3d58ee5f70ae63d73d7658d4026f2eefd2f204c81682cb7 which is generated from an empty recovery phrase (“”) using the Parity wallet. There have been 8772 transactions on this address with a total of 5215.586 Ethereum transferred. This address is very active and being monitored for inbound transactions which are immediately transferred out by one of many private key holders watching this address. The below figure (Figure 13) illustrates a monthly transaction count since this address was first used on the Ethereum blockchain in March of 2017.

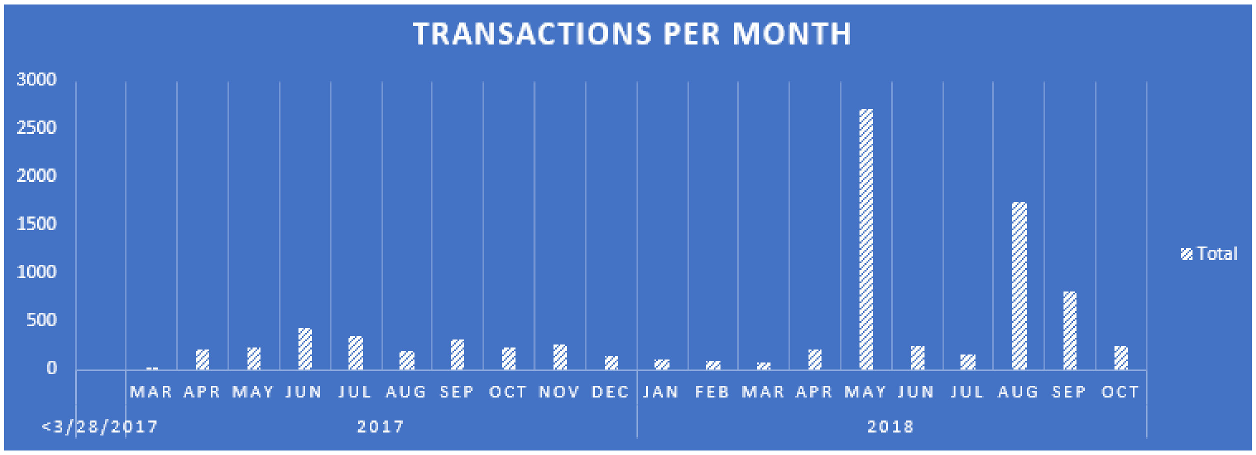


Figure 13. Transaction volume over a “” Passphrase parity wallet.

Ethereum Address Corresponding to Null Public Key

The Ethereum address of 0xdcc703c0e500b653ca82273b7bfad8045d85a470 (PubHashEmptyStr) is derived from a keccak256 hash performed on an empty “” string, instead of an ECDSA public key[5]. This address has had 815.643 ETH transferred to it over 28 transactions spanning roughly 3 years that is irrecoverably lost.

All-Zero Ethereum Address

The Ethereum address 0x0000000000000000000000000000000000000000 (EthAddrNull) has had Ethereum erroneously transferred to it. It is all but certain that this was due to user or code error where a null destination address was provided. This address has had 7289.472 Ethereum transferred to it over 1169 transactions spanning roughly three years that is irrecoverably lost.

[4] https://etherscan.io/address/0x4c636a08fdf3692a9bca111e8a7f4a0e28eb4457

[5] https://twitter.com/cyrus/status/978775247334748160?lang=en

Conclusion:

Through the enumeration of select areas in the 256-bit private key space where errors are likely result in the use of weak keys, we have discovered 49,060 transactions spread over 732 public keys for which we have the corresponding private keys, with a total transfer amount of over 32 Ethereum.

Additionally, we found other addresses that exist and have been erroneously used when transferring Ethereum, the most damaging, in terms of ETH potentially exposed to theft, is the Parity derived Ethereum address that is based on an empty string passphrase (e.g. “”). This address has had 5215.586 Ethereum transferred to it which as of March 2019 is worth 730,182 USD. Any Ethereum transferred to this address can be transferred out by individual in possession of the Ethereum private key that was derived from an empty Parity passphrase.

Moreover, there have been a combined total of 8,105 Ethereum transferred to the following two addresses:

- 0x0000000000000000000000000000000000000000 (PubAddrNull)
- 0xdcc703c0e500b653ca82273b7bfad8045d85a470 (PubHashEmptyStr)

The above addresses had over 1197 transactions spanning over three years. Any Ethereum transferred to these addresses are irrevocably lost as there are no known private keys that correspond to them. It is beyond current computing resources to try to enumerate private keys in order to locate the correct corresponding public keys. It is worth noting that in a large-scale event where large sums of Ethereum are erroneously lost or transferred it may be possible to lobby the Ethereum governing body, made up of node operators, to perform a hard fork. That is, to create a new version of the blockchain that would reverse any effects of the error. One such event happened in 2016 when a hard fork was performed on the Ethereum blockchain to reverse the effects of the DAO attack [3].

At the height of the Ethereum market, when 1 Ethereum had a market value of approximately 1432.00 USD, on January 13 2018, the combined losses of either using erroneous destination addresses or using wallets derived from weak or known keys amounted to 13,319 Ethereum valued at $18.8 million, and the address behind the active hacking campaign we stumbled onto, at the height of the Ethereum market was holding $54.3 million.

Due to the popularity and easy monetization of cryptocurrencies combined with the evidence that there are highly successful hacking campaigns ongoing to steal these virtual currencies, it should be concluded that any systems that handle private keys will be at an increased threat for targeted attacks. Software developers that design software or systems that interact with highly valuable private keys should incorporate all available defense in depth principles to counter present threats and use innovative measures to counter advanced present and future threats against these high value assets.

Future Work:

Any implementation that uses public key signing based on ECDSA or similar cryptographic algorithms can be examined for key generation errors. The obvious examples would be other blockchain implementations such as Bitcoin, Waves, Ripple, ZCash, Monero, among others.

It may also be worth exploring source code repositories and/or analyzing Android/iOS/Windows applications for use of hard coded private keys. Additionally, there may be cases where weak or manually generated test keys used on test blockchain networks may have migrated to production code.

While the legality of transferring crypto currency tokens from addresses that are stumbled upon through the enumeration of private keys is uncertain, it may be worth exploring activities surrounding the keys discovered in our research. However, it appears that the pilfering or camping on a private or group of private keys to watch inbound transactions and immediately issue an outbound transaction is very profitable (at least in one instance we identified). Investigating the interactions between poorly generated private keys and their destinations may reveal patterns that may allude to larger scale activities that are currently not well understood or documented. Law enforcement or individuals wishing to pursue civil action to reclaim crypto currency tokens may have an interest in such research or tools to help identify patterns and destination addresses used by individuals or organizations involved in the theft of these virtual currencies.

References

[1] "etherscan.io," [Online]. Available: https://etherscan.io/.

[2] "etherscan.io," [Online]. Available: https://etherscan.io/chart/address.

[3] "wikipedia," [Online]. Available: https://en.wikipedia.org/wiki/The_DAO_(organization).

[4] D. G. Wood. [Online]. Available: https://ethereum.github.io/yellowpaper/paper.pdf.

[5] S. S. Stuart Haber. [Online]. Available: https://www.anf.es/pdf/Haber_Stornetta.pdf.

[6] J. Stretch, "packetelife.net," [Online]. Available: http://packetlife.net/blog/2010/nov/23/symmetric-asymmetric-encryption-hashing/.

[7] J. Stanley, "TechSpot," [Online]. Available: https://www.techspot.com/article/1567-blockchain-explained/.