Mathematical Weapons of Wallet Penetration

by

Mehraj Parouty[1]

Vulnerabilities in cryptocurrency(crypto) wallets, which were once only theoretical possibilities, are historical lessons that we can learn from, in 2020. Cryptocurrencies worth several Billions of USD have been reportedly stolen to this date. On the 4^{th} of August 2020, for example, Bitfinex Exchange posted a $400 Million reward to recover 120,000 of its stolen Bitcoins, worth around $1.4 Billion. From backdoors in hot wallets to side channels in cold wallets, the ingenuity of hackers always seem to be, at least, one step ahead of security experts.

While the traditional method of stealing currencies online is phishing for passphrases, PIN codes, private keys and etcetera, additional means exist to steal cryptos. These means are "Mathematical weapons of wallet penetration"; which this paper aims to describe. The paper first describes the traditional method of phishing for passphrases and private keys that secure a wallet. Second, Mathematical means of compromising crypto wallets is explored. And, last, we discuss dis-incentives for stealing cryptos.

Traditional Pick-walleting: Phishing

Phishing is probably the easiest way of gathering private key information to gain control of a wallet; although the scope of the attack is often limited to individual victims. A wallet is essentially a key pair; a public bank account number (or IBAN) and private passphrase for a FIAT wallets or a private key and a public key for a crypto wallets. The owner of the private key owns whatever assets that are in the wallet. Bitcoins, autoexecutable lines of codes (such as smart contracts on the Ethereum

[1]E-mail: linuxrules@protonmail.com

blockchain), digital content (such as those on Steem blockchain) and even online identities (such as the identity system of Selfkey) are assets that can be stored in a wallet over which the owner of the specific private key has full and exclusive control.

Poisoning individual computers

Phishing exploits for private passphrases or the private keys, thus, allow an attacker to, not only, gain access to the wallet but to control it as well. Exploitation tools such as MSF Venom or BeEF can easily and freely be installed on most linux distributions. BeEF, a cross-site scripting framework, also comes pre-installed in Operating Systems such as Kali Linux; with the executable in the folder:

```
/usr/share/beef-xss
```

A javascript file is used as a hook to tie the client's browser to the BeEF server. This file can be embedded in an html webpage by simply adding a single line of code such as

```
<script src="http://<IP>:3000/hook.js"></script>
```

In this script, the port is 3000 and the IP is the attackers IP address. In order to remain anonymous, proxychains and port forwarding can be used. In Kali Linux, proxychains can be activated by simply modifying the built-in proxchains configuration file, found in

```
/etc/proxychains.conf
```

Commenting the random chains line by adding a hashtag and uncommenting the line with dynamic chains by removing the hashtag is all that needed to set up proxychains. Further anonymity can be implemented by the attacker's MAC address using macchanger. So, an anonymous attacker can gain full access to a users browser that contains his/her private passphrase or key and consequently gain full control over the user's wallet.

For this reason, wallet providers often add 2-factor authentification or One-Time-Passwords that are communicated to a mobile or another device. This requires phishing on two separate devices in synchrony and is harder. Other means of adding difficulty to the phishing task have

also been implemented by banks such as Lloyds TSB in the UK which has enforced a random digit passphrase system. Users logging in are asked to enter 3 characters from their passphrase (which is more than 12 characters long) that are in an automatically generated random order. So, an attacker can only phish for 3 acaracters of the passphrase at a time. An attacker would then be required to repeat the phishing a sufficient numer of times in order to find the passphrase. While all of these are additional layers of difficulty, they cannot hamper a malicious attack.

Poisoning entire libraries

Attackers have also reportedly injected popular libraries with malicious codes and backdoors; infecting thousands of computers at a time. Because such libraries are open source and consequently scrutinised for bugs by hundreds of developers on a daily basis, one would assume that injecting a malicious code would not go unnoticed. However, opportunities to compromise such libraries are not inexistant but rare and a historical reality.

Event-stream library is a javascript package with more than 2 million weekly downloads on the npmjs.com repository. When the original package maintainer sought a new developer to continue the package development, an opportunity of injecting a backdoor had presented itself to the new package maintainer, going by the handle Right9ctrl. The latter immediately issued an update to Event stream v 3.3.6 with a new dependency called Flatmap-stream which was where the backdoor resided. Bitpay's Copay App was using this library and their users information and private keys were communicated to the copayapi.host URL on port 8080. Both Bitcoin and Bitcoin Cash were stolen.

Beyond Phishing: Mathematical Exploits

Apart from phishing for private keys, crypto wallets are also vulnerable to mathematical exploits. In order to understand how wallet addresses are secured, let us explore the Mathematical techniques of cryptography, through which they are generated. In the case of Bitcoin wallets, two cryptographical techniques are particularly important for security, namely:

1. Hash (SHA256)

2. Elliptic Curve Cryptography (ECC)

Hash SHA256

Secure Hash Algorithm (SHA) is a set of cryptographic hash functions designed by the United States National Security Agency (NSA). Hash functions are all mathematical functions that can map data of arbitrary sizes to data of a fixed size [1]. A cryptographic hash function is a special class of hash functions that possesses the following properties:

1. it is deterministic so the same message always results in the same hash
2. it is quick to compute the hash value for any given message
3. it is infeasible to generate a message from its hash value except by trying all possible messages
4. a small change to a message should change the hash value so extensively that the new hash value appears uncorrelated with the old hash value
5. it is infeasible to find two different messages with the same hash value

To put it in simple terms, the cryptographic hash function takes in any kind of digital information as input to result (output) in an alphanumeric string; such as, for example: 0xe3b0c44298fc1c149afbf4c8996fb92427 ae41e4649b934ca495991b7852b855. This function allows one to easily verify that some input data maps to a given hash; but it is deliberately difficult to reconstruct the input data by knowing solely the hash. This function has 2 characteristics:

1. **Unequivocal:** the hash (output) is like the fingerprint of the input data. From the hash of a digital input, one can't create the original digital input but from the input, the hash can be easily created.
2. **Collision Resistant:** nobody should be able to find two different input values that result in the same hash output. In other words, for any different input, there will always be different outputs.

These allow one to check for **data integrity** by comparing the computed "hash" (the output from execution of the algorithm) to a known

and expected hash value. For example, computing the hash of a downloaded file and comparing the result to a previously published hash result can show whether the download has been modified or tampered with.

Elliptic Curve Cryptography

Elliptic Curve Cryptography (ECC) is the Mathematics that protects your wallets and is used to generate a public key from the private key. ECC is an approach to public-key cryptography (watch video) based on the algebraic structure of elliptic curves over finite fields, such as:

$$y^2 = ax^3 + bx + c \tag{1}$$

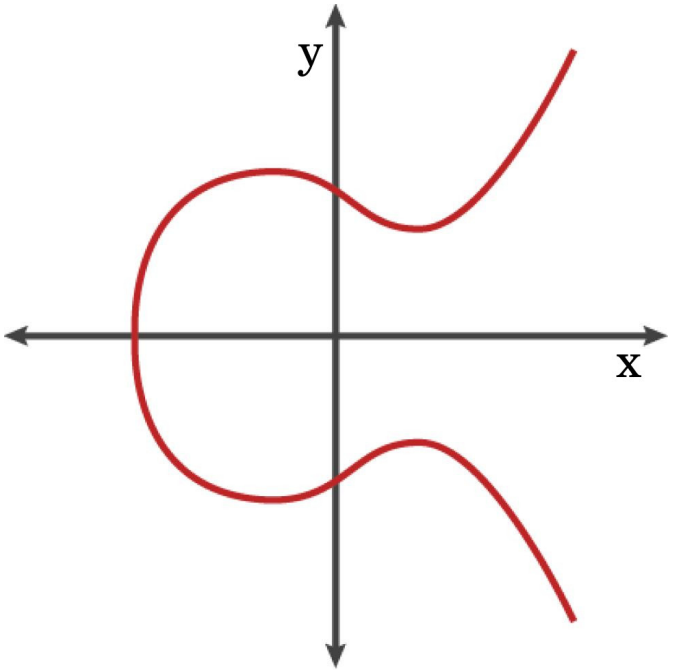

Figure 1: Example of an Elliptic Curve.

The elliptic curve for Bitcoin, for example, is specified by the secp256k1 standard with $a = 1$, $b = 0$ and $c = 7$; that is

$$y^2 = x^3 + 7 \tag{2}$$

The secp256k1 standard has a pre-specified point on the curve which is called the generator point, G and is always the same for all keys. Your

private key, on the other hand, is in the form of a randomly generated number, say **p**. This is then point multiplied to G to produce another point somewhere else on the curve, sayg **K**. That is:

$$K = p * G \tag{3}$$

where
p is the private key,
G is the generator point, and
K is the resulting public key, a point on the curve.

Because the generator point is always the same for all Bitcoin users, a private key p multiplied with G will always result in the same unique public key K. The relationship between p and K is fixed. However, while **it is computationally easy and straightforward to move from p to K, it is computationally difficult and nearly impossible to move from K to p.**

In order to understand why your private key, p, cannot be derived from K, your public key, through division, it is important to note that the mathematical operations of **elliptic curve point doubling (ECDouble) and elliptic curve point addition (ECAdd) are not the same as arithmetic doubling and addition** in the traditional sense. Point addition of two points, F and G on an elliptic curve means the following operations:

1. Draw a line between F and G
2. Find the point on the curve where the line intersects
3. Reflect that point across the x axis

Therefore, in the case where $F = G$, point addition means that F is added to G in the limit that F tends to G. Under this limit, the line joining the two points is a tangent line at the point G. In this case, as is the case for Bitcoin, the public key, K, is obtained by adding the generator point, G, to itself p times. Adding G to itself, first, requires finding where the tangent at G intersects the curve. This point is, by convention, called $-2G$. Reflecting this point across the x axis, we get the point $2G$. Therefore, ECAdd, the operation of adding a point to itself in Bitcoin wallets, is equivalent to drawing a tangent line on the Generator point, G and finding where it intersects the curve, and then reflecting that point across the x axis. And this operation is repeated p

times to find $K = pG$, the public key. The following diagram shows the process for deriving $2G$,$4G$ and $8G$ as a geometric operation on the curve with $p = 8$.

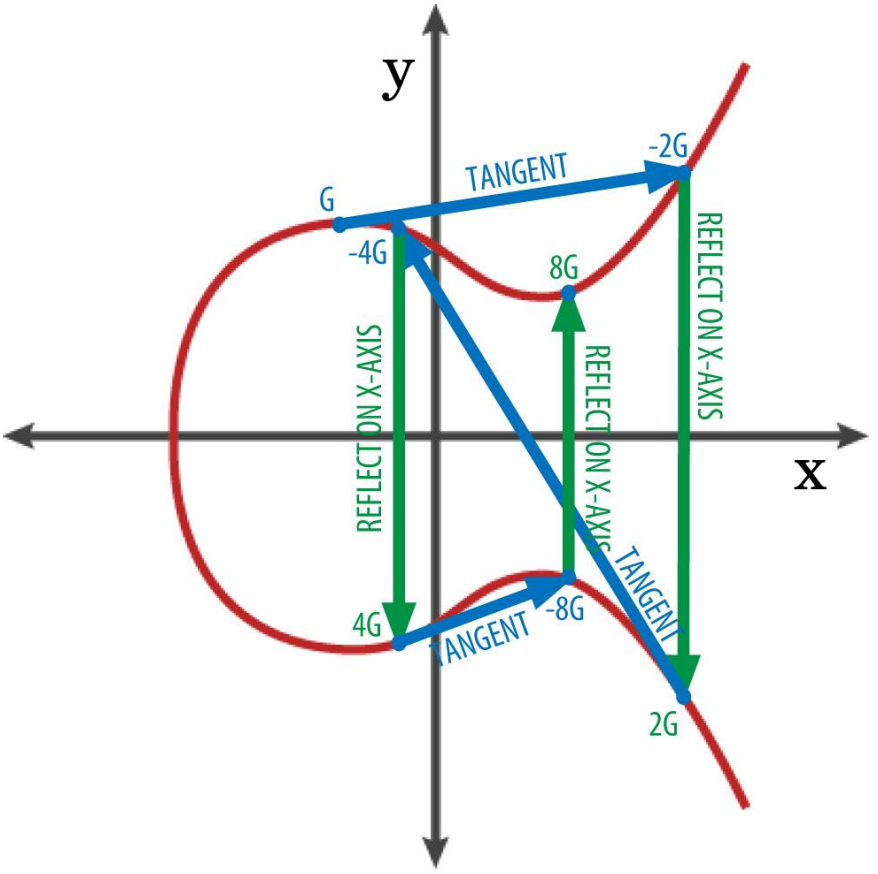

Figure 2: Process for deriving 2G, 4G and 8G from G as a geometric operation on the curve from the book Mastering Bitcoin.

Note that this is unequivocal. Given the point G, and given $p = 8$, it is easy to reach the point $K = 8G$ on the curve. Starting with G, the blue tangent and green reflection arrows give point $2G$, and performing this operation again gives the point $4G$ and $8G$. So, knowing the private key, p, the public key, K, can be found easily. However, given the point G and the point $K = pG$, it is computaionally difficult to find $p = 8$.

Generating a Public Address

Now that we have the basics, let us see how a Bitcoin Address is created.

1. The 1^{st} step is to have a reliable source of randomness and get a random number (256-bit)
2. Next, we apply a SHA256 to that number and get the private key (p). [Unequivocal Process]
3. By using elliptic curves, we obtain the public key (K), as explained above.
4. Then, SHA256 and RIPEMD160 (a different hash function) is applied and we finally get the Bitcoin Address. [Unequivocal Process]

We can represent the address in many different formats. The most common is with a QR Code.

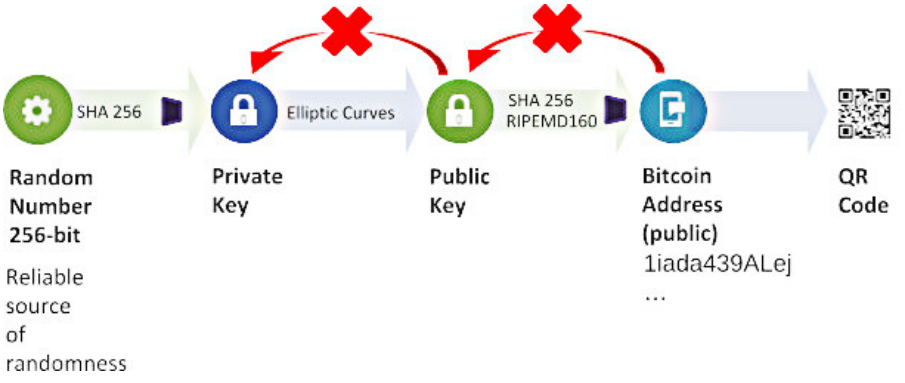

Figure 3: Process of generating a Bitcoin public address from RSK Educate.

Data based mathematical attacks

While it is difficult to infer a private key, p from a public key, K, it has been historically possible to retrieve the private key via other routes. We recall that the 1^{st} step in generating a wallet address is to have a reliable source of randomness to get a random number. However, this reliability has often been compromised.

In order to validate and warrant blockchaining through a consensus algorithm such as Proof-of-Work, Elliptic Curve Digital Signature Algorithm (ECDSA) is used to transform a message and a private key into a signature. An ECDSA signature, s, is composed of 2 parts and looks like follows:

$$s = \left(r, \frac{z + r.p}{k} \right) \tag{4}$$

where $r = (k.G)_x$ with
$k =$ a random number,
$G =$ the global curve base point,
$p =$ the private key, and
$z =$ the hash of the signed message.

In the above, r, s and z are published on the blockchain. G is known. So, knowing k allows the private key, p, to be known through simple algebraic manipulations. This has been the case in some browsers that used an unupdated version of the function math.random().

Other ways of finding k in network devices also exist [2]. When k follows a pattern and is not randomly generated, a lattice attack, is possible. When a large number of people use the same k, basic algebra again makes it possible to infer k. When two messages use a compromised random number generator to produce the same k for both transactions, then k can again be algebraically derived. And, knowing k implies knowing p, the private key. These ways were first pointed out in the 27^{th} Chaos Communication Congress on PS3 epic fail. Cryptocurrencies such as Bitcoins have been stolen in this manner.

Side channel based mathematical attacks

The attacks mentioned above, all, rely on an internet connection to transmit information to and from the victim. One might assume that chopping off a wallet from its internet connection would prevent information leak. The so-called cold wallets, i.e. wallets that are not connected to the internet, are indeed the safest wallets available. However, "safest" and least hackable does not mean that cold wallets are unhackable. In side channel attacks, information is not leaked through direct access to data but through indirect means; such as gauging memory chips for different voltage outputs with different PIN inputs using Mathematical algorithms.

The security research team at Ledger hardware wallet, Donjon, has a history of finding vulnerabilities from its competitors; such as Trezor, Coinkite, Shapeshift and others. On December 2018, Donjon team disclosed an attack to the security team at Trezor; attack which works on several hardware wallets, namely B Wallet, Keepkey, Trezor One, and Trezor T. On December 2019, Donjon team again published an article with high-level details of an attack on Shapeshift's Keepkey wallet.

KeepKey uses a protected storage, where the private keys are encrypted by a storage key which can be unwrapped by the user PIN. The PIN verification is implemented using the function:

`storage_isPinCorrect_impl`

This function is used to derive the wrapping key from the pin, wraps the storage key that is stored in flash, hashes that storage key and finally compares this hash with the key fingerprint stored in flash. The PIN verification mechanism for Keepkey wallet is as follows:

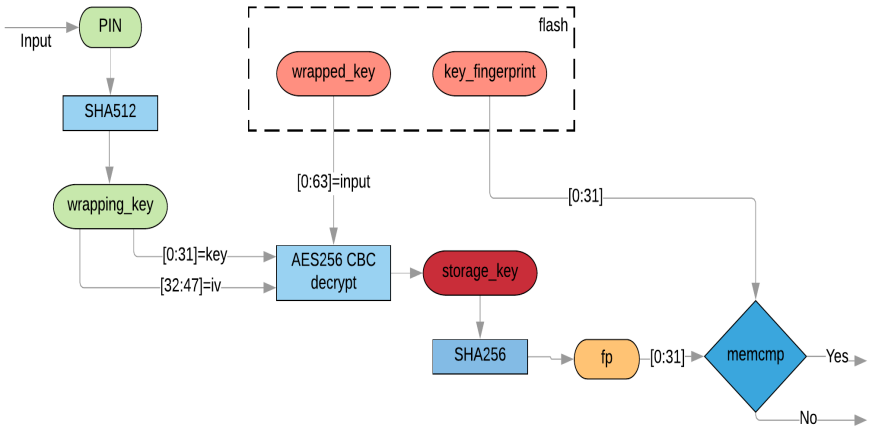

Figure 4: KeepKey PIN verification mechanism from Donjon Tech Blog

Donjon used their self-developed Lascar tool to analyse the code that is running on the chip and their self-created Scaffold evaluation board for the actual side-channel attack. The mathematical algorithm then pools different voltages for each PIN that is cryptographically mixed with the secret data that is stored in memory. A database showing how each of these secret values compares to its voltage output is then established; and this allows the PIN to be recoved.

Discussion

The Mathematics of cryptography is essential to secure digital wallets. The reason that a wallet is secure is because bruteforcing a private key

from a public key would necessitate computations that are highly entropic. Entropy is a measure of possibilities [3] and the higher the entropy, the more difficult it is to extract the private key and, consequently, the more secure is a cryptographically based wallet. Cold wallets, with no internet connection, are the most secure wallets. However, their security is not absolute but relative to other wallets. Private keys can still be retrieved.

Coldcard MK2 wallet, for example, can be hacked using a so-called "fault-injection attack". This attack causes a glitch in a computer that triggers a deugging mode. In this mode, the guess limit is not active and an attacker can launch a bruteforce attack. However, in order to cause the glitch, a high-powered laser is directed onto a specific location of the chip with a specific timing. This Laser fault injection rig costs roughly $200,000 and also requires special skills to operate. So, when it costs around 20 Bitcoins to hack a wallet that possibly has less than 20 Bitcoins, attackers are dis-incentivised; at least, attackers who adhere to rational choice theory [4], although there is scarcely any protection against a motivated attacker with sufficient resources.

References

[1] Jean-Sébastien Coron et al. "Merkle-Damgård revisited: How to construct a hash function". In: *Annual International Cryptology Conference*. Springer. 2005, pp. 430–448 (cit. on p. 4).

[2] Nadia Heninger et al. "Mining your Ps and Qs: Detection of widespread weak keys in network devices". In: *Presented as part of the 21st {USENIX} Security Symposium ({USENIX} Security 12)*. 2012, pp. 205–220 (cit. on p. 9).

[3] Pornin T. "Deterministic usage of the digital signature algorithm (DSA) and elliptic curve digital signature algorithm (ECDSA)." In: (2013) (cit. on p. 11).

[4] John Von Neumann and Oskar Morgenstern. *Theory of games and economic behavior (commemorative edition)*. Princeton university press, 2007 (cit. on p. 11).