Cover the keys, lah!

thecryptofruit, December 2022

More than we care to admit, our lives are bound to a few bytes of data that represent our passwords or our keys, first with the rise of the internet and now with web3. How to keep cryptographically protected things actually secure? Here are some thoughts that I've assembled primarily for my own sake, hopefully it might help you, too.

Intro

Any part of a person's digital presence, any access to email, socials or work documents, any NFT or cryptocurrency somebody holds, all of them are protected by some kind of password, or a private key for crypto-tokens. What is the optimum way to keep these keys *per se* is an extremely interesting question, but too complex for this piece as it has more social connotations than technological[1]. Instead, here I try to explore the **overall security considerations of the custody of crypto assets** that are somewhat organizational in nature.

There are two main principles to keep at heart and one can't over emphasise their importance:

Not your keys, not your coins.
&
Security is a chain, strong only as its weakest link.

If you don't understand why they are to be treated seriously, you haven't done your homework and at this point would be advised to take them as axioms.

We would bother with the really pesky and inconvenient things that follow below because it does matter - too many people have already lost their money and many more will, lots of businesses have permanently shuttered their operations and we simply wouldn't want to experience such a catastrophe first hand. Keep in mind these assets are publicly discoverable on blockchain and our setup is being poked at non-stop, without us even knowing - that is a guaranteed, permanent and unavoidable threat.

[1] The answer is not as simple as putting the passwords in a password manager, as it just means that now we're reducing the problem to protecting the "master password" or something to the password manager. It's also not as simple as putting cold wallets and recovery seeds in a safety box or a vault, as these also require a key or other restricted access. I want to stress that all this always narrows down to protecting a few secret characters and it takes a good look into social patterns and historical perspective to address it in fullness. Humans have protected their most precious things since forever and in quite astonishing ways.

Quick primer on terminology

There are lots of confusing and ambiguous terms in the web3 space, we're not doing a good job at all. As one of those who are allergic to what "crypto" or a "token" means these days, I'm pointing a finger to utterly poor and overzealous marketing, but am aware that it is what it is and so let's work with it.

Password - we can shield private data by encrypting it with a secret key, a.k.a. a password: the same secret key (with limited transformations) is used for encrypting and decrypting the data, thus if we share the shielded message between Alice and Bob, they both need to know that key. This is called symmetric key cryptography, e.g. AES.[2]

Private key - part of the keypair in asymmetric cryptography that is kept private and can (i) prove the ownership of the public key (sign a transaction) and (ii) decrypt a message that was encrypted with corresponding public key. Note that while Bitcoin uses ECDSA to create keypairs, it doesn't encrypt anything, instead it heavily utilizes digital signatures.[3]

Public key - part of the keypair that can be shared without revealing the protected data, but usually has detrimental privacy implications.

Address (in the context of blockchain) - is a derivation of the public key, in Bitcoin it is equal to the public key and in Ethereum is the last 20 bytes of the Keccak-256 hash of the public key

Account - a structure in account-based blockchains, such as Ethereum, and contains: address, balance, code, nonce, storage pointer. An account can be an EOA (externally owned account) which is private key-based, or a contract account, which doesn't have a private key behind it, but is addressable and has a code that manages access.

Wallet - an interface to the account(s) that provides connectivity to the blockchain network via a node. A *cold wallet* is a way of storing private data on an off-line medium, such as steel or paper. A *hot wallet* is software that has access to private keys, is connected to the internet and interacts with the blockchain network (usually indirectly via a centralized web3 provider). It's not hard to guess that the majority of wallets are hot wallets and that MetaMask leads the pack in the popular Ethereum network.[4]

---

[2] *Encryption(msg,key)=ciphertext; Decryption(ciphertext,key)=msg*

[3] Encrypting messages: *E(msg,pubKeyBob)=c; D(c,privKeyAlice)=msg*
Verifying messages: *Sign(msg,privKeyAlice)=signature; Verify(signature,pubKeyAlice)=mgs*

[4] MetaMask is a browser extension that accesses the encrypted private keys on the local device. I repeat, a browser extension - not a very safe place. By default it connects to Infura node provider, a part of Consensys umbrella. And tracks IP addresses, likely linking them against wallet addresses.

Node - a peer in the blockchain network that does many things, but primarily broadcasts transactions and other messages. Running a node is resource consuming, so most people use "public" web3 providers, such as Infura, Alchemy, Tenderly etc on Ethereum, but it's rather straightforward, if inconvenient, to run a node locally, either on a regular PC or a dedicated, pre-build device like Dappnode.

Custody - the obvious, yet not appreciated, thing about custody is that it is about custody or ownership of crypto-assets, i.e. who has access to private keys. *Custodial* wallets have managed private keys, which means a user does not have a direct access to the keys and relies on the legal system for protection. With *non-custodial* wallets users can access private keys directly and are thus in full control. *Self-custody* is sometimes referred to as non-custodial wallets. Note that with hardware wallets, such as the popular Trezor and Ledger Nano devices, the user is in full control of the private keys, but the keys are actually never exposed outside the hardware, so you can't actually "see" them - which is quite neat, because you can't accidentally lose something you don't have/see; and if the device gets stolen, it's likely still safe.

Generating private keys - remember that private keys are the critical part: to hold the keys in a secure way, the keys must also be generated in a secure way, otherwise the attacker could recreate them. It is such a delicate procedure that it should be only done if we really know what we are doing. It all starts with a source of randomness (entropy), everything is based on this so it is an extremely delicate operation. With the entropy, we can then derive a password, a private key, or, in deterministic-wallets, a mnemonic (a set of 12-24 words) that is more memorable and easier to write down than funky characters. A mnemonic is used to create a seed via key stretching and finally this seed can be used to derive child keys which are the basic approach of organizing multiple keys that share a common root[5]. This process must be done with maximum care! In Qubes OS, you can quickly create a new, disposable virtual machine that was never online, do the work i.e. generate the keys and safely transfer them to an always offline vault, then discard the virtual machine and with it all the traces of the work completed.

Splitting access - having multiple, separate parts to gaining access is beneficial in many cases (i.e. not putting all eggs in one basket) and there are two levels at which we can split the access/authorization:

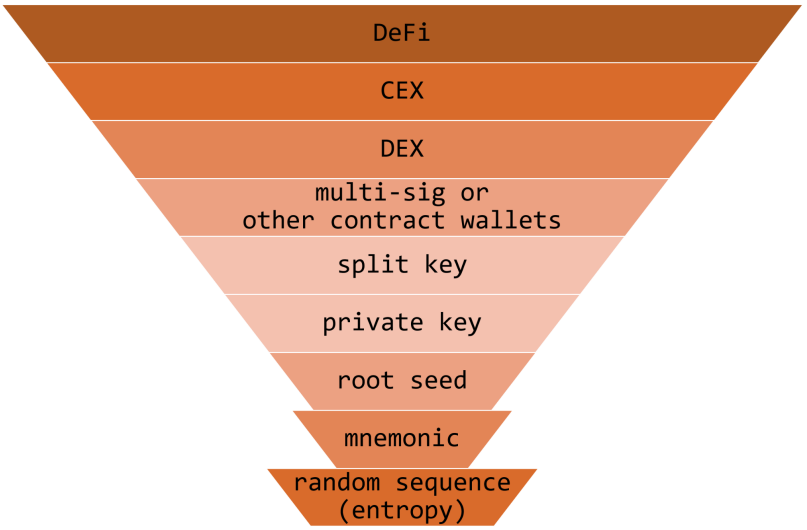
- we can either split the computations "over" a private key, so that multiple parties participate in the signing of a transaction (see *MPC*, secure multi-party computation, or the lesser *SSS*, Shamir's Secret Sharing)[6],
- or we can require multiple "normal" participants to contribute in the signing with "normal" private keys (*multi-sig*, which is network-specific, e.g. it is an

[5] Here's an excellent walk-through keys and wallets: https://github.com/bitcoinbook/bitcoinbook/blob/develop/ch05.asciidoc
[6] https://blog.bitgo.com/multi-sig-vs-mpc-which-is-more-secure-699ecefc8430

OP_CODE in Bitcoin[7], but in Ethereum requires a custom multi-sig contract implementation[8]).

MPC vs multi-sig

A simplified inverse pyramid shows the distance (complexity) from the assets as we store them in various options currently available. It's a burden to manage the bottom of the pyramid - it gives us the **most freedom on account of most responsibility**! The more layers the pyramid has, the quicker it can top over. On the other hand, the upper layers enable more features, e.g. can be used in exchanging assets or even generating extra yield.



Which one do you choose? How do you create it and initialize it, how do you store it and how do you use it?

[7] https://en.bitcoin.it/wiki/OP_CHECKMULTISIG
[8] A common library to use in 2022 is: https://app.safe.global/
Many years ago, it was Parity's library, however after a horrible hack it went away: https://www.parity.io/blog/the-multi-sig-hack-a-postmortem

Note 1: Do not do your own cryptography, as tempting as it might be, it will be utterly bad, guaranteed[9]. It is also futile to do "extra" steps like scrambling mnemonics[10] and it is often a nuisance going with the latest in the making[11].

Note 2: If you are generating the entropy yourself, do everything you can in order to have a secure process. As a first rule, do not do it with a device that is connected to the internet, or will be ever in the future, and do a clean wipe of all the components when done with the generation and the resulting data is safely moved to a secure location. Qubes OS is your friend in this in the sense of providing a reasonably clean, yet handy environment.

Note 3: With the rise of MPC-based solutions (used by the biggest custody providers like Fireblocks and Coinbase, but can also be DIY[12]), we still need to take care of managing secret data, and the challenges of secure generation and distribution of keys still remain. There's no free lunch - more complexity, more risks.

Security considerations

All standard security principles apply with crypto assets, but the situation here is actually worse: an exposed weakness immediately and irrevocably drains the funds, which implies we must focus on preventative security controls and, ultimately, on physical controls, such as protecting access to private keys in the most direct, physical way.

Secure custody covers two types of threats: **malicious** and **accidental**. That is, we must protect against outside attacks, but also against human mistakes, software errors and such. These dimensions are orthogonal, though it bears noting that their final outcome is the same - the assets becoming permanently inaccessible to us - whether as a mistake that opened us to an attack, or an accident that locked or burned the assets. In almost all cases there is no going back.

Weaknesses

Weaknesses or **vulnerabilities** here have primarily to do with private keys, passwords that protect them, and to a lesser extent, malleability[13] of transactions we make.

---

[9] Just one of the crazy examples out there: https://dci.mit.edu/research/2019/3/31/cryptanalysis-of-curl-p-and-other-attacks-on-the-iota-cryptocurrency

[10] BTCRecover can descramble quickly: https://btcrecover.readthedocs.io/en/latest/BIP39_descrambling_seedlists

[11] Ethereum hashing function is "almost" SHA3: https://ethereum.stackexchange.com/a/554

[12] "Awesome" list on MPC by Dragoș Rotaru: https://github.com/rdragos/awesome-mpc and Boston University's MPC resources page: https://multiparty.org/

[13] https://en.wikipedia.org/wiki/Malleability_(cryptography)

Threats

These weaknesses have the potential to be exploited and so present a **threat**. Threats come across multiple dimensions.

1. *Endogenous vs exogenous threats*

Endogenous (us making mistakes) or exogenous (attackers, or third-party: e.g. outsourced software and hardware we use or external providers we subscribe to); here we should mention one special case of an exogenous threat that is called Advanced Persistent Threat (APT), a state-backed actor that has practically infinite resources and against which we can at best prolong the defense a little bit, knowing that ultimately our setup will be compromised.

2. *Temporal threats*

Longevity of the setup - what is secure today, might not be tomorrow or in few years, because a part of our setup would become obsolete due to maintenance challenges or become insecure due to general technological progress, but also we ourselves could become incapacitated, thus our crypto assets would come under risk.

3. *Location-based threats*

Location of keys/passwords - are they stored locally on our computer or at a remote location or in a cloud; does this location have access to the internet and can thus leak; do we maintain a redundant storage for backup and if so, all previously mentioned considerations apply for the backup again and independently.

Keys as sensitive data

We can apply standard data protection controls to private keys, because keys are data, thus consider *data at rest* (where is it stored, is it encrypted, what permissions are set up), *data in transit* (once we access the keys, we have to move them somewhere, e.g. input them into a wallet, which might be online, thus consider the security of transportation), *data in use* (the keys are present in the computer memory, consider how secure is our operating system, hardware, etc.).

Responding to threats

A standard threat response process applies:
preparation - identification - containment - post-incident activity.

N.b. once we detect that an incident has happened, we must move whatever assets are left to a safe wallet - with haste and make sure to provide the transfers with extremely high transaction fees, so our transactions beat those of the attacker! Every second counts (blockchain transactions take between a couple of seconds to a couple of minutes). Since we must have assets spread across different locations and at various layers in the technological stack, it is our decision which ones we'll urgently move, but we should usually move all at a particular layer. Example: the attack happened at one of our MetaMask accounts, thus we won't be moving our

cold storage assets, but we will move all remaining assets in all MetaMask accounts because we suspect there was a browser hijack or a browser extension compromise at play.

However, detecting that we've been hacked is not straightforward. Imagine owning tens or hundreds of different accounts for various assets across multiple blockchains. How do we know that an unauthorized transfer was made? We'd be among a minority if we at least maintained an up-to-date overview of our whole portfolio, let alone have a complete notification and alerting system in place.

Friendliness of security

User experience in accessing crypto-assets is usually inversely related to security.[14] We are getting better, no doubt, but the basic premise remains unchanged. Here's why, in my experience at least. User experience has to do with how easy it is for users to understand and use something to achieve their goal. Alas, by making something simple to the user, we usually mask complexity away, but while it now looks simpler on the outside, we have also just increased complexity by introducing a wrapping layer - and created another attack vector. Secondly, by making it easy to use, we removed friction, enabling faster access with less hops and less clicks in less time - but with no friction, in all likelihood, we are now increasing our chances of making mistakes (remember the irreversibility of blockchain transactions), which doubles if we think that if something is easy, we might do it more often and more is not always better. Lastly, UX improvements often involve additional action steps and/or additional people or services, which is detrimental at least as much as it is beneficial to our capability of performing autonomous actions[15].

Inversely related to the UX is the **closeness of private keys**. **Hot wallets** have private keys available in an app that is connected to the internet, which means that signing transactions is a matter of clicking a button. Their only protection is that the private key is somehow encrypted, e.g. on our disk or in the cloud, and is decrypted once we enter the app and hopefully enter a PIN, a password or something not too dramatic. **Cold wallets** on the other hand have private keys (or mnemonics or seeds) in an offline storage or sometimes there is no app at all, just a private key written on a paper, engraved in steel or similar. Having such an extra

---

[14] As bad as it sounds and with a plethora of "services" providing "easy to use" wallets, I am getting a feeling this is more of a fundamental principle, rather than an opportunity for a fancy product. I've been closely involved with access management for about two decades, half of it related to crypto-assets, and I've seen a lot of progress, a lot of approaches, and a lot of too-good-to-be-true marketing commotion.

[15] This is a contentious issue. My reasoning is that less savvy users should also be treated as first class citizens, always having a path out of the challenges they face, rather than introducing opaque "solutions" under pretense of user-friendliness. Being autonomous in transacting is important for humans, because we know that slavery is unacceptable. In order to be autonomous, one must be able to act, regardless of *any* context, e.g. wealth, race, location etc. At this moment, this is *only* possible via the blockchain-based networks that the reader here might take for granted. There is not one single such possibility with legacy systems. Thus, to preserve this uniformity of accessibility, UX and other upper layers must not act as dependencies and must allow graceful degradation.

gap from the private keys provides better protection, but is cumbersome, so we would use cold wallets only for infrequent operations. **Warm wallets** are something in between: they keep private keys in an online environment, but require human interaction either at signing transactions or by running software locally.

Is something really more secure if it's offline, or in a safety box or we're running the wallet software locally? In principle, yes, the attack surface is much smaller, but in practice it depends. Many people have lost their paper cold wallets, or have forgotten their password to an offline disk or old computer with private keys on. In other words, the number and severity of exogenous threats increases with our distance from private keys, but decreases for endogenous.

Evaluating risk

Which finally brings us to evaluating our risk exposure. *Risk* is measured by the impact of an exploit on our system and the likelihood of the threat to be exploited. There are ways to quantify these.[16] The relation between the impact and likelihood indicates an important lesson: we need to take a balanced approach to protecting crypto-assets, being most careful where the impact of an attack is highest, just like our bodies have brains much better protected than, say, hair.

General setup

By now it should be clear that the system to protect our private keys should in majority of cases be balanced, it should hedge against different threats, be usable and friendly as much or as little as it can be, and should be safe not just in this moment, but hopefully for a few years. In an extreme, but a literal sense, some parts of this system must endure even if our computer crashes (it will, eventually), even if the wallet/encryption software we use gets deprecated (it will, eventually), and it is worth considering our own incapacitation, which will happen one day, and its consequences to the accessibility of our crypto-assets.

The setup chosen will depend on case by case - consider *how often* we will need access to *how big* a part of our portfolio. The objectives of a trader are very different to the objectives of someone wanting to maximize long-term security.

Top to bottom, we usually have:

- a small part in a hot wallet for daily things that won't hurt if lost;
- a significant chunk in one or more of these: a hardware wallet, a warm wallet, a multi-sig setup or a smart wallet, which all require an additional manual step, sometimes involving another person, and while it would hurt if lost, it won't break us;
- a major chunk goes to cold wallets, where we have flexibility in spreading across storage (multiple locations, multiple physical types of cold wallets,

[16] For a comprehensive paper on this, I recommend NIST's Guide for Conducting Risk Assessments, which is available online.

key-splitting, and potential encryption at this level, though be extremely careful not to over-complicate and get locked out);
- when swapping assets, trading, staking or similar actions, we usually must use a centralized platform (thus losing direct control over the assets) or a decentralized protocol (usually can maintain some control), so a good practice is to minimize the time there: do the action and withdraw back.

If we're using a backup system for anything of the above, we have to treat it with the same care as the original.

Few tips from practice

We should exchange our experiences in order to improve everyone's setup, however we will keep noticing that each of the setups is unique. What I have and what my objectives are, is similar but not exactly the same to yours. Security is tailored-made.

In no particular order, lots of these were learned the hard way …

Using different kinds of wallets:

- Store access credentials for the hot wallet in a reasonable way, but keep the recovery codes in a safe place - chances are you'll need to re-create it from the seed, e.g. when changing computers or even reinstalling the browser.
- Organize the multi-sig with a 2-of-3, having two keys handy (e.g. in the password manager, a hardware wallet, a mobile app, or a hot-wallet account), and the third key stored tucked away, say, where storing the cold wallet as we'll only need it in an emergency.
- Prefer a hardware wallet to a hot one whenever possible.
- Hardware wallets need their firmware and apps updated from time to time. Before doing big, delicate transactions, do check your version against the latest release, take the time.
- Multi-sig in practice takes a lot of time, esp. if we have to chase the signatories across time zones. Don't underestimate the coordination effort.
- Safely store the recovery notes, i.e. if they're on paper, protect it against both fire and water.

Keeping it sane:

- Keep track of the custody setup in one way or another, but consider this document compromised.[17]
- Software and hardware deprecate in time. Wallets get deprecated, too, who remembers Armory, once considered the ultimate wallet? Consider making a snapshot of the current working environment that can be reliably re-run in the future.

[17] I am at a loss to understand why so many people are still unaware of an almost 150 years old Kerckhoffs's principle: https://en.wikipedia.org/wiki/Kerckhoffs%27s_principle

- Wallets have different “standards” they follow. E.g. BIP-39 “Mnemonic code for generating deterministic keys” is as standard as it gets, however an excellent wallet like Electrum is not 100% onboard[18].
- Separate personal operations from work! Segregate the environment for each project/organization! Qubes OS makes this a walk in the park.[19]
- Do not put all cold-wallets and recovery mnemonics of other wallets into the same place. Treat all locations with the same high level of scrutiny. It is inconvenient, but that is the nature of this.

Whatever setup we have, I can guarantee some rough edges and nervous situations. Keep in mind that this is *still* a young field and a rapidly changing technology - very few things here are stable.

Live with it

After we have all things set up, we need to keep an overview and be alerted if something unpredictable happens. Here are a few useful tools that can save us from tears and sweat:

- Rotki - portfolio tracker that is open-source and aims to preserve privacy
- OpenZeppelin Defender Sentinels - alerts on the events/transactions
- Tenderly Alerting - easy alerting
- DappNode - run your own node(s) on your own hardware and point your Rotki or other local apps to use it, very easy to use and very web3 friendly

Recommended reading

Wallet Security: The ‘Non-Custodial’ Fallacy[20] is an excellent exploration into what is custody, really. Written by Nassim Eddequiouaq and Riyaz Faizullabhoy from *a16z crypto*, you know it’ll be good. As I’ve explained above in more detail, we can start our setup from various levels, from generating private keys on our own, to fancy smart wallets. The more layers in the stack we use, somewhat easier the UX, but greater the attack surface.

How to Back Up a Seed Phrase[21] is an article by a veteran Jameson Lopp from Casa and has some great general tips, I esp. liked the list of approaches and services for seed storage.

An In-Depth Look at the Parity Multisig Bug[22] offers some important insights into smart contract security. Written by Lorenz Breidenbach, Phil Daian, Ari Juels, and Emin Gün Sirer from Cornell’s IC3, it offers some juicy details about one of the largest hacks ever, this one affecting a multi-signature library that was at the time

---

[18] Electrum has some concrete details on it, nonetheless it was made 2 years prior to BIP-39.
[19] This is as it should be done always, for so many reasons:
https://www.qubes-os.org/news/2022/10/28/how-to-organize-your-qubes/
[20] https://a16zcrypto.com/wallet-security-non-custodial-fallacy/
[21] https://blog.lopp.net/how-to-back-up-a-seed-phrase/
[22] https://hackingdistributed.com/2017/07/22/deep-dive-parity-bug/

"the industry standard". I would esp. like to raise their remark about how complexity is security's oldest enemy, which obviously applies for young technologies at least as much as anywhere else.

Leaderboard on Rekt[23] is a list of hacks in this space, which should provide us with more than enough motivation to treat custody with the necessary seriousness. Also a humbling reminder that we all make mistakes and should keep learning and improving.

Outro

As Jameson Lopp would say in a not so subtle way[24]:

Whatever setup we have for protecting the private keys and passwords in general, a good approach is to have a balanced system, segregated across threats and use-cases, and keeping an open eye on that ultimate little thing that unlocks it all!

[23] https://rekt.news/leaderboard/
[24] https://twitter.com/lopp/status/1590796484714188801