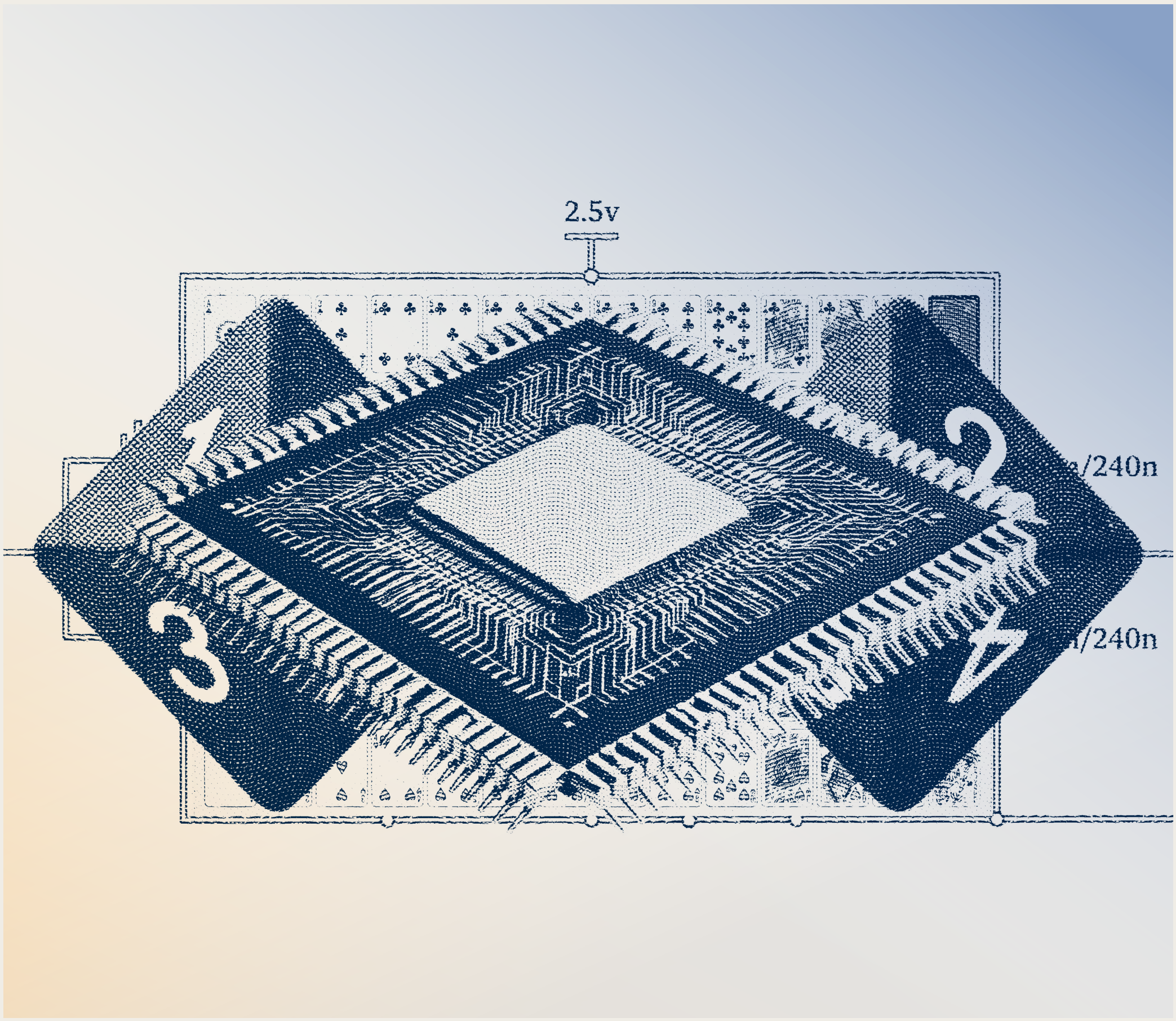
The randomness question at the heart of your *hardware wallet*

ENTROPY, RNGs, KEY CREATION, AND GUARDING AGAINST DEVICE VULNERABILITIES



TOM HONZIK

Table of Contents

I. Introduction

II. Random numbers and entropy

III. Computer RNGs

IV. Hardware wallet chips and TRNGs

V. Manually creating your own key

VI. Does manual entropy eliminate trust in a hardware wallet?

VII. Selected primary sources

Introduction

When your hardware wallet generates a key to protect your bitcoin, it must do so in a way that is sufficiently random. Otherwise, the security of the key (and any funds it protects) is permanently compromised from the beginning. Using a technological device to generate a bitcoin key is one of the highest-trust, least-understood actions regularly taken in the bitcoin economy. Users generally do not understand what processes are occurring within the device while a key is generated, and have instead deferred to the reputation of the manufacturer, or have become comfortable upon hearing “it’s open source.”

This worked fairly well for years, but justifiably fell under heavier scrutiny after the July 2026 Coldcard vulnerability. Coldcard’s source code was publicly viewable, and the company's reputation was widely seen as strong, prior to the vulnerability discovery. As usual, hindsight is 20/20. Yet during the five years that the bug existed, the keys generated by most Coldcard devices were not sufficiently random at all, causing many people to lose substantial amounts of bitcoin when this became known.

The solution to this dilemma isn’t to expect all hardware wallet users, advertisers, and wallet application providers to independently conduct expert-level security research into the codebase and components of hardware wallets, which may not succeed in catching every bug. The solution is to distribute risk and eliminate single points of failure with multisig wallets (which proved strong even while involving Coldcards) and to consider incorporating multiple separate technologies for the keys within the multisig arrangement.

No matter your security model, better understanding key generation techniques is a valuable exercise. If you’re using a single key to protect bitcoin, knowing how the key was created can be critically important. If you are using multiple keys in a multisig arrangement, learning whether different hardware utilizes overlapping techniques can be useful information. In this article, we’ll discuss the basics of random number generation, the processes used by some popular hardware wallets, and how keys can be created manually for greater direct involvement.

Random numbers and entropy

A bitcoin key is a large number. When creating a key, the goal is to select a number that is so large, and so random, that another person or computer ever guessing it is practically impossible. Entropy quantitatively measures how well the process that produced the number achieves this goal of security, and serves as a foundational concept of modern cryptography (outside of bitcoin as well).

Does randomness even exist?

Creating a large number isn't difficult, but creating one randomly can be tricky. Whether true randomness exists at all is a question that extends into philosophy. Some people believe that everything can be reduced to cause and effect, so if enough information is known about relevant variables, the outcome can be accurately predicted with careful calculation. For example, when you flip a coin, if the angle and velocity of the coin is known, along with the features of the surrounding environment, knowing how it will land is possible before it actually occurs.

Studies have shown that the human mind isn't very good at producing randomness either, due to certain biases—such as avoiding repetition and round numbers that "feel less random," or overcorrecting upon learning about such biases. Picking numbers in your head is considered an insecure method of creating cryptographic keys.

Tasking a computer with generating unpredictable, "random" numbers has been a difficult problem as well. Computers were primarily designed to execute functions accurately and reliably without variance. Running the same inputs through the same functions or formulas should result in the same outputs every time, otherwise the computer would be seen as unreliable. Therefore, creating a random output would first require a random input or random function. That means the goal of producing randomness requires randomness to already be present, forming a circular problem without an obvious solution.

Whether or not true randomness exists, a practical goal is to create sufficient randomness, such that others can't predict the outcome. When you flip a coin, the result might not be truly random, but when done privately outside of a laboratory, the results are unpredictable in practice because the inputs aren't precisely measurable. Guessing how the coin will land essentially has a 50% success rate (1 in 2, or 1 bit of entropy). Guessing how a series of 256 coin flips all landed in the proper order will have a success rate of 1 in 2^{256} (256 bits of entropy).

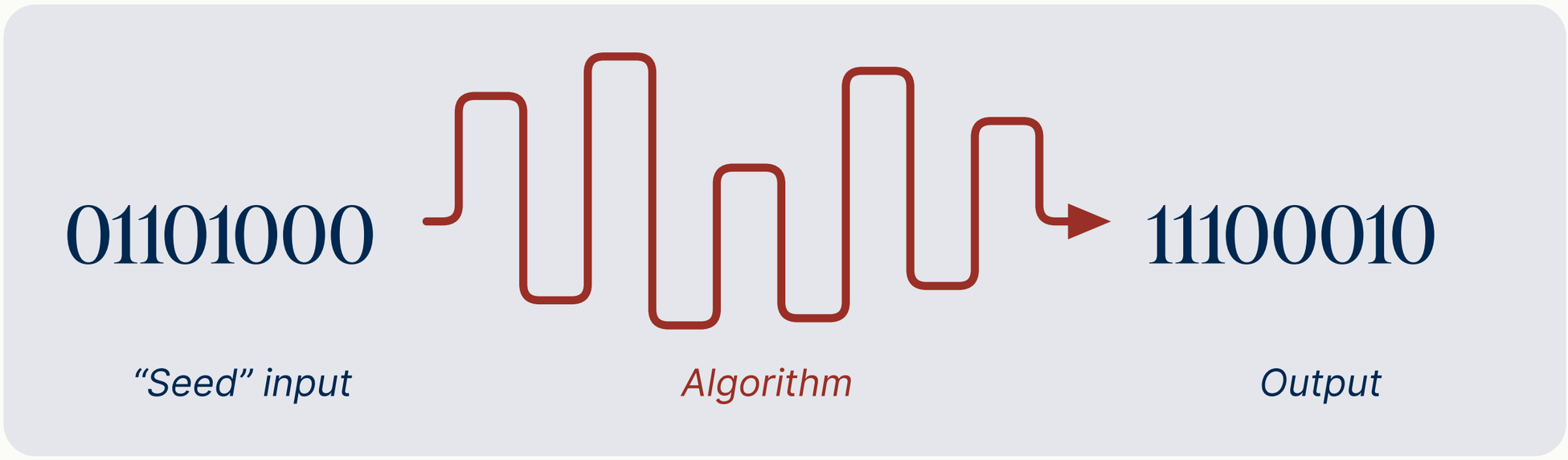
Computer RNGs

When using a computer or phone, you may notice instances of supposed randomness, such as while playing a game (like Tetris or blackjack) or shuffling a music playlist. These instances may be utilizing functions called RNGs (random number generators). As discussed in the previous section, getting computers to create randomness has been a challenge. There are two main categories of RNGs: pseudorandom number generators (PRNGs) and true random number generators (TRNGs).

Pseudorandom number generators

The prefix “pseudo” means “fake,” immediately implying that PRNGs are not quite random. A PRNG takes an input and runs it through a mathematical algorithm, transforming it into an output. A good PRNG will tend to produce a uniform distribution of outputs—for example, if it’s used to produce numbers from 1 to 6, it would be like rolling fair 6-sided dice where each outcome is equally likely, rather than loaded dice. Outputs will also not have a clear pattern. To an uninformed observer, it can seem quite random.

However, if the starting input (or “seed”) becomes known, the results are entirely deterministic and predictable. Running the same input through the same algorithm will always yield the same result. Famed mathematician and physicist John von Neumann once joked, "anyone who considers arithmetical methods of producing random digits is, of course, in a state of sin."



Therefore, PRNGs by themselves are not a good choice for creating secret cryptographic keys for important purposes such as protecting bitcoin. They require a starting value from somewhere, and if that value isn’t already sufficiently random, the output of the PRNG will be guessable and insecure.

PRNGs can be quite useful for less important purposes such as video games or shuffling a music playlist, where the user doesn’t really care about the details so long as it seems to behave randomly in practice. PRNGs are efficient, using minimal resources to create many outputs quickly, which is less true for TRNGs.

True random number generators

For serious applications where PRNGs alone are insufficient, TRNGs offer an alternative. As we discussed above, the word “true” being used to describe randomness can raise objections, but this is the adopted terminology nonetheless. A TRNG is fundamentally different from a PRNG, because it doesn’t rely on a seed and algorithm. Instead it gathers physical data and uses that to produce the output, similar to flipping coins or rolling dice, but at a micro-scale within the device electronics. If the physical data is unpredictable, then it can be a strong source of randomness.



There are several methods for gathering unpredictable data to implement a TRNG. Brief explanations of some of the more popular ones that might be used for key generation, are as follows:

1. **Ring oscillator jitter:** A tiny electronic loop turns on and off as fast as it can. Heat and other electrical noise creates unpredictable variance in the exact timing of each flip, which is then observed and recorded as data. This method is very inexpensive and readily available with ordinary circuit parts, leading to its widespread use. A potential weakness is that it can be disrupted by malicious close-range signals.
2. **Diode avalanche noise:** A small electronic part called a diode is pushed into a controlled breakdown state, causing unpredictable bursts of electric current. This method has a stronger, less fragile signal than ring oscillator jitter, but is less readily available inside general-purpose equipment, making it a specialist's choice.
3. **Radio static reuse:** A chip with radio hardware listens to background static that's influenced by unpredictable thermal noise in the system. This method may be readily available for devices that include a radio function, but must have a fallback if the radio is turned off.

Are TRNGs trustworthy?

Hardware TRNGs were introduced into personal computers by Intel in 1999. Before that, randomness was attempted with software scraping entropy from the environment, often asking users to type random keystrokes or move their mouse, from which it would record factors such as the precise timing of the movements.

In the following years, TRNGs began to be included within pretty much every computer and smartphone, and they have become widely relied upon for security applications. Despite decades of widespread use, physical TRNG noise circuits have an exceptionally strong security record.

No confirmed major bitcoin loss has been traced to a TRNG itself failing or being manipulated. TRNG exploitation has occurred (e.g. Markettos & Moore, CHES 2009), but within a laboratory environment. Key-generation failures (e.g. Android SecureRandom, Milk Sad, Coldcard) have all stemmed from software problems surrounding the random number generator, never from an actual TRNG failing or being manipulated. In the Coldcard case, for example, a bug in the device's firmware caused it to silently bypass its own hardware TRNG.

Hardware wallet chips and TRNGs

Hardware wallets across the bitcoin industry use a variety of techniques and internal equipment to generate keys. Some hardware wallets combine multiple sources of entropy, which are then mixed together using something like a hash function. Combining multiple independent sources of entropy in this manner can be advantageous, because the overall entropy will be at least as strong as the strongest source, not the weakest source. For example, if three independent entropy sources are properly mixed, and then it's discovered that two of the sources were flawed and predictable, as long as the third source contributed sufficient entropy, the overall entropy is also secure. When implemented correctly, multiple independent sources add redundancy.

Most hardware wallets utilize at least one computer chip with a TRNG as an entropy source. Each computer chip may or may not be a secure element, a component built specifically to be tamper-resistant. Secure elements help protect against attackers who might gain physical access to the device and try to use sophisticated machinery to extract the key from the hardware.

In the following table, we examine some of the most popular bitcoin hardware wallets, and what is publicly documented about their entropy sources and computer chip choices at the time of writing. Some of the chips have official certifications about the randomness produced by their TRNGs, and secure elements may also have a physical certification rating for their tamper-resistance.

Entropy sources in bitcoin hardware wallets

DEVICE	LIFECYCLE / FIRMWARE	SOURCE CATEGORY	SOURCE / COMPONENT	MAKER	TYPE	PHYSICAL ENTROPY MECHANISM	INTRODUCED	PHYSICAL CERTIFICATION	RANDOMNESS CERTIFICATION
Trezor Model One	Discontinued — supported	Hardware entropy	**STM32F205**	STMicroelectronics	MCU	Ring oscillator jitter	2010	No certification	No certification
		Other input	**Computer / phone entropy**	—	External input	—	—	—	—
Trezor Model T	Discontinued — supported	Hardware entropy	**STM32F427**	STMicroelectronics	MCU	Ring oscillator jitter	2013	No certification	No certification
		Other input	**Computer / phone entropy**	—	External input	—	—	—	—
Trezor Safe 3	Current	Hardware entropy	**STM32F429**	STMicroelectronics	MCU	Ring oscillator jitter	2013	No certification	No certification
		Hardware entropy	**OPTIGA Trust M V3**	Infineon	Secure element	Undisclosed	2022	CC EAL6+	AIS 31 PTG.2
		Other input	**Computer / phone entropy**	—	External input	—	—	—	—
Trezor Safe 5	Current	Hardware entropy	**STM32U585**	STMicroelectronics	MCU	Ring oscillator jitter	2021	No certification	NIST E11
		Hardware entropy	**OPTIGA Trust M V3**	Infineon	Secure element	Undisclosed	2022	CC EAL6+	AIS 31 PTG.2
		Other input	**Computer / phone entropy**	—	External input	—	—	—	—
Trezor Safe 7	Current	Hardware entropy	**STM32U5G**	STMicroelectronics	MCU	Ring oscillator jitter	2024	No certification	NIST E116
		Hardware entropy	**OPTIGA Trust M V3**	Infineon	Secure element	Undisclosed	2022	CC EAL6+	AIS 31 PTG.2
		Hardware entropy	**TROPIC01**	Tropic Square	Secure element	Undisclosed	2025	No certification	No certification
		Other input	**Computer / phone entropy**	—	External input	—	—	—	—
Ledger Nano S	Discontinued — no updates	Hardware entropy	**ST31H320**	STMicroelectronics	Secure element	Undisclosed	2015	CC EAL5+	AIS 31 PTG.2
Ledger Nano X	Current	Hardware entropy	**ST33J2M0**	STMicroelectronics	Secure element	Undisclosed	2019	CC EAL5+	AIS 31 PTG.2
Ledger Nano S Plus	Current	Hardware entropy	**ST33K1M5**	STMicroelectronics	Secure element	Undisclosed	2021	CC EAL6+	AIS 31 PTG.2
Ledger Stax / Flex	Current	Hardware entropy	**ST33K1M5**	STMicroelectronics	Secure element	Undisclosed	2021	CC EAL6+	AIS 31 PTG.2
Ledger Nano Gen5	Current	Hardware entropy	**ST33K1M5**	STMicroelectronics	Secure element	Undisclosed	2021	CC EAL6+	AIS 31 PTG.2
Coldcard Mk2 / Mk3	Discontinued — **FINAL FIXED FW 4.2.0**	Hardware entropy	**STM32L475 / L496**	STMicroelectronics	MCU	Ring oscillator jitter	2016 / 2017	No certification	No certification
		Hardware entropy	**ATECC508A / 608A**	Microchip Technology	Secure element	Undisclosed (508A); ring oscillator jitter (608A)	2015 / 2018	No certification	NIST E46 (608A only)
		Other input	**User entropy**	—	User-provided entropy	—	—	—	—
Coldcard Mk4 / Mk5 / Q	Current — **FIXED FIRMWARE REQUIRED**	Hardware entropy	**STM32L4S5VIT6**	STMicroelectronics	MCU	Ring oscillator jitter	2017	No certification	No certification
		Hardware entropy	**ATECC608B / 608C**	Microchip Technology	Secure element	Ring oscillator jitter	2020 / 2023	No certification	NIST E46
		Hardware entropy	**DS28C36B**	Analog Devices	Secure element	Undisclosed	2017	No certification	No certification
		Other input	**User entropy**	—	User-provided entropy	—	—	—	—
BitBox02	Current	Hardware entropy	**ATSAMD51J20A**	Microchip Technology	MCU	Undisclosed	2018	No certification	No certification
		Hardware entropy	**ATECC608B**	Microchip Technology	Secure element	Ring oscillator jitter	2020	No certification	NIST E46
		Other input	**Factory secret**	—	Provisioned secret	—	—	—	—
		Other input	**Computer / phone entropy**	—	External input	—	—	—	—
		Other input	**Password hash**	—	User-derived input	—	—	—	—
BitBox02 Nova	Current	Hardware entropy	**ATSAMD51J20A**	Microchip Technology	MCU	Undisclosed	2018	No certification	No certification
		Hardware entropy	**OPTIGA Trust M V3**	Infineon	Secure element	Undisclosed	2022	CC EAL6+	AIS 31 PTG.2
		Other input	**Factory secret**	—	Provisioned secret	—	—	—	—
		Other input	**Computer / phone entropy**	—	External input	—	—	—	—
		Other input	**Password hash**	—	User-derived input	—	—	—	—
Foundation Passport Core	Discontinued — supported	Hardware entropy	**STM32H753**	STMicroelectronics	MCU	Ring oscillator jitter	2017	No certification	No certification
		Hardware entropy	**Avalanche circuit**	Foundation	Analog RNG circuit	Diode avalanche noise	N/A	No certification	No certification
		Hardware entropy	**ATECC608A**	Microchip Technology	Secure element	Ring oscillator jitter	2018	No certification	NIST E46
Foundation Passport Prime	Current	Hardware entropy	**SAMA5D28**	Microchip Technology	MPU	Undisclosed	2015	No certification	No certification
		Hardware entropy	**Avalanche circuit**	Foundation	Analog RNG circuit	Diode avalanche noise	N/A	No certification	No certification
		Hardware entropy	**ATECC608C**	Microchip Technology	Secure element	Ring oscillator jitter	2023	No certification	NIST E46
Blockstream Jade Classic	Current	Hardware entropy	**ESP32**	Espressif	Wireless SoC	Radio-assisted analog noise	2016	No certification	No certification
		Other input	**Power and temperature readings**	—	Sensor input	—	—	—	—
		Other input	**Timing and device state**	—	Runtime data	—	—	—	—
Blockstream Jade Core	Current	Hardware entropy	**ESP32-S3**	Espressif	Wireless SoC	Radio-assisted analog noise	2020	No certification	No certification
		Other input	**Factory secret**	—	Provisioned secret	—	—	—	—
		Other input	**Computer / phone entropy**	—	External input	—	—	—	—
		Other input	**Power and temperature readings**	—	Sensor input	—	—	—	—
		Other input	**Timing and device state**	—	Runtime data	—	—	—	—
Blockstream Jade Plus	Current	Hardware entropy	**ESP32-S3**	Espressif	Wireless SoC	Radio-assisted analog noise	2020	No certification	No certification
		Other input	**Factory secret**	—	Provisioned secret	—	—	—	—
		Other input	**Computer / phone entropy**	—	External input	—	—	—	—
		Other input	**Camera imagery**	—	Sensor input	—	—	—	—
		Other input	**Power and temperature readings**	—	Sensor input	—	—	—	—
		Other input	**Timing and device state**	—	Runtime data	—	—	—	—
Block Bitkey	Current	Hardware entropy	**EFR32MG24**	Silicon Labs	Secure MCU	Undisclosed	2022	PSA Certified Level 3	No certification

An assortment of chip manufacturers

As we can see from the table above, the hardware TRNG components responsible for generating entropy in popular hardware wallets can be manufactured by a variety of companies. Understanding how these manufacturers overlap could be useful for someone seeking equipment diversification, such as while choosing hardware wallets for a multisig wallet.

- **STMicroelectronics** is a European multinational company and a popular choice for microcontroller (MCU) TRNGs that are used within all Trezors, all Coldcards, and the original Foundation Passport. They also produce the certified secure element TRNGs used within all Ledger devices.
- **Microchip Technology** is an American company that produces MCU/MPUs used within BitBox02 models and the Foundation Passport Prime. They also produce secure elements used within Passport models, Coldcard models, and the original BitBox02.
- **Infineon Technologies** is a German company that produces certified secure elements for Trezor Safe models and the BitBox02 Nova.
- **Espressif Systems** is a Chinese company producing the chips that contribute entropy to the Blockstream Jade lineup of devices.
- **Silicon Labs** is an American company producing the secure MCUs used in the Block Bitkey.

For every chip in our list, the full details surrounding the silicon and TRNG mechanisms are proprietary, and can't be verified or audited by a regular consumer. Secure elements in particular are often especially secretive for security purposes—even the category of TRNG mechanism is typically undisclosed. The Trezor Safe 7 uniquely includes a TROPIC01 secure element designed by Trezor's sister company, Tropic Square (both under the umbrella of SatoshiLabs). This chip is substantially more transparent than conventional secure elements, with much of its architecture publicly available. However, the underlying TRNG block is not open source or publicly auditable, so some trust in proprietary silicon remains necessary.

Unique design approaches of hardware wallet manufacturers

The earlier chart reveals that all the hardware wallet manufacturers covered have made unique decisions that might be interesting while evaluating your options.

BitBox combines the most diverse set of entropy sources, for a total of five. There are two hardware TRNGs, a hash of a user-input password, a static device-unique random number generated at the factory, and entropy generated by a companion app on the user's computer. Assuming correct implementation, this means that an attacker would have to find flaws in all of these separate sources to rebuild the overall entropy and find the private keys. Similarly, Trezor and Jade also include entropy through their companion app on the user's host computer or phone, which could serve as a last line of defense if the internal device TRNGs were impacted by a vulnerability discovery.

Ledger has taken the opposite approach, opting for simplicity by using entropy from just one hardware source. Their secure element of choice has an official physical certification and randomness certification. Bitkey also uses one source from a secure MCU.

Foundation devices are unique in their inclusion of a diode avalanche noise circuit, which is built with ordinary discrete components and an open-source schematic that regular consumers can investigate, as opposed to the TRNGs within computer chips. This creates a materially more transparent source of hardware entropy.

Blockstream has intentionally avoided a conventional secure element to make their devices inexpensive and practical for users to build independently. They use an uncertified general-purpose MCU with a radio-assisted TRNG mechanism. As an alternative protection against attackers with physical access to the device, they've implemented the option to encrypt the wallet with an external blind oracle, which they've described as a "virtual secure element."

Manually creating your own key

Instead of using a hardware wallet's complicated and often opaque entropy generation techniques, it's also possible and relatively straightforward to create a key entirely from your own entropy, taking matters into your own hands. This can be done by flipping coins, rolling dice, shuffling a deck of cards, or other similar activities. With correct procedural steps, the results will be sufficiently random, and can be converted into seed phrase words to import the key into a hardware wallet.

First, generate sufficient entropy

Bitcoin keys are most commonly built with either 128 bits or 256 bits of entropy, either of which are secure options. A bit is a binary digit, 0 or 1. Flipping a fair coin 128 times and recording the results will create 128 bits of entropy. Rolling a 6-sided die (D6) can also be converted into bits by various methods, as can dice with a different number of sides, or classic playing cards.

If your entropy will be used for creating a secure key, it's important to ensure that the process is done carefully. Dice must be fair, not loaded. Cards must be verified as a complete set, and shuffled sufficiently, without the assistance of a machine.

The process must be done privately, outside of the view of people or cameras, and without verbally speaking the results near microphones.

Everything must be recorded offline and remain offline. Never alter your results "so that it looks more random," as this could introduce human bias that actually does the opposite and makes the results less random. This is not a comprehensive list of all possible considerations—if you choose to use one of these methods, it's important to think adversarially and ensure your entropy is random and secret.

Second, convert the entropy into BIP39 words

Seed phrases are a standardized way (BIP39) to represent your bits of entropy as words, which helps avoid mistakes while transferring the data. The official BIP39 wordlist contains 2048 unique words, each of which represents a unique set of 11 bits ($2^{11} = 2048$). For example, 10110000001 is represented by the word "rabbit," and 11000010001 is represented by the word "season."



A lookup table (example 1, example 2) can enable you to easily convert your bits of entropy into official seed phrase words. It's recommended to avoid actively using a lookup table on an online computer or phone, because scrolling to the location of your words might give clues to an attacker who has managed to track your screen. Instead, use a printed physical copy.

Seed phrases are typically 12 or 24 words, depending on whether 128 or 256 bits of entropy are used. The mathematically inclined might notice that things don't quite add up: 12 words each representing 11 bits equals 132 bits total, not 128. This is because the final 4 bits are not entropy, but a checksum calculated based on the 128 bits of entropy, which helps to ensure the seed phrase is recorded without errors. 24 words represents 264 bits, with the final 8 bits serving as a checksum on top of the 256 bits of entropy.

Finally, find the last seed phrase word

The final word of a 12-word seed phrase consists of 7 bits of entropy and a 4-bit checksum, while the final word of a 24-word seed phrase consists of 3 bits of entropy and an 8-bit checksum. If you've already generated the full entropy, you just need to calculate the checksum, but unfortunately this isn't easy to do by hand or even with a typical calculator. The process involves a SHA-256 hash function with many complicated mathematical steps that must be executed perfectly, and shouldn't be attempted by an individual for anything important. Instead, it can be done quickly and accurately with a computer.

However, the computer must have all of the entropy in order to calculate the checksum, and it's also important to keep the entropy permanently secret. Anyone who finds your entropy can also calculate your seed phrase and your associated bitcoin keys. Therefore, typing your entropy into a conventional computer or phone that can connect to the internet is dangerous. Even if the device is temporarily offline, it could have a malicious keylogger tracking your entries that later sends the data externally once it's back online. Therefore, the best type of computer to use for this task is once again a hardware wallet, which is designed to protect sensitive key information from online exposure.

Among the hardware wallets covered in this report, BitBox, Passport, Jade and Coldcard can assist with providing valid final words, after entering the first 11 or 23 words during a seed phrase import. At the time of writing, Trezor and Ledger require the user to provide a full set of 12 or 24 words, and are less helpful for this procedure. Bitkey isn't a conventional hardware wallet and is restricted to its own custody system; it doesn't allow seed phrase imports at all.

The Foundation Passport randomly selects the final word, contributing the remaining 3 or 7 bits of entropy from the device itself. BitBox, Jade and Coldcard display valid final words (2^7 or 128 possibilities for a 12-word seed phrase, and 2^3 or 8 possibilities for a 24-word seed phrase) and let you choose. You could choose randomly with your own method, or use the final bits of entropy you generated previously to determine the selection, which Blockstream suggests in a process called "correcting the final word." There will only be one valid word from the word list that begins with the same 3 or 7 bits as what you generated.

Does manual entropy eliminate trust in a hardware wallet?

The phrase "don't trust, verify" is popular among bitcoin enthusiasts. It serves to discourage blind trust in others, especially when it comes to managing scarce, valuable assets like bitcoin. However, completely eliminating trust is practically impossible. No one possesses the expertise to independently, flawlessly audit every line of code and every physical component involved with their bitcoin custody setup. Taking this idea to the extreme, even someone with comprehensive knowledge must have learned what they know from resources produced by other people.

Therefore the realistic goal is not to eliminate, but to minimize trust in any single person, group, software, or hardware. Choosing a hardware wallet with a certified TRNG or multiple independent sources of entropy can reduce the chance of a flawed process, but doesn't eliminate the possibility of vulnerabilities in the surrounding implementation, which is exactly what occurred in the Coldcard incident. Similarly, manually generating your own entropy reduces your exposure to possible hardware wallet flaws (Coldcard users who imported manual entropy properly were protected), but doesn't completely eliminate risk after importing your key into the device.

Hardware wallets are used for sensitive operations beyond merely producing entropy and generating private keys. If the private keys will be used for securing bitcoin, the hardware wallet must share public key information externally, without ever exposing a private key. For most users, this implies trusting the device to do this safely. Furthermore, if you receive bitcoin and later want to spend it, once again the hardware wallet will be trusted to sign your spending transaction and share the signature externally, without exposing any private keys.

Other notable trust-minimization features

Trust in a hardware wallet to communicate externally without exposing secrets can be reduced with air-gapped communication, a feature offered by some device models. MicroSD card transfers or QR scanning can reduce exposure to attacks targeting continuously connected interfaces such as USB or NFC, although transaction details must still be verified on the hardware wallet's screen.

When a hardware wallet signs a transaction, it must use an unpredictable number called a nonce. This nonce is often produced deterministically from an unpredictable source, such as the private key and transaction hash, rather than from a TRNG. However, if the device has a flawed nonce, perhaps due to malicious firmware, it can covertly leak the private key to attackers while signing transactions. This theoretical attack has not been seen in the wild, but was demonstrated particularly efficiently in the 2024 "Dark Skippy" disclosure. Mitigation can be achieved by using a nonce partially constructed from a source outside the device. Jade makes this technique available in a feature they call anti-exfil, and BitBox has a similar feature called anti-klepto.

Features such as these are interesting, nuanced ways to further reduce trust in a hardware wallet. The degree to which a hardware wallet is open source is another relevant factor. However, the easiest and cleanest way to minimize trust in a hardware wallet is to use a multisig security arrangement, so that your bitcoin can't be accessed without multiple independent keys. That way, any one hardware wallet failing can't lose bitcoin.

Redundancy with multisig is the strongest protection

A hardware wallet flaw can put bitcoin at immediate risk if the funds are secured in a singlesig arrangement. Whether or not the hardware wallet was used to create the key, the device is still tasked with protecting the key in its entirety while communicating with external tools.

A singlesig arrangement means that the single key is all that's needed to access the bitcoin—a single point of failure. This is unavoidable with a singlesig model, even when using additional techniques such as passphrases or Shamir's secret sharing.

Using a multisig arrangement fundamentally shifts the trust away from complex device software and hardware, to the extremely simple, battle-tested multisignature implementation built into the bitcoin protocol itself. Verifying that a multisig wallet was set up correctly is far more manageable than verifying that a hardware wallet is flawless.

If a hardware wallet generates an insecure key or leaks key information during operation, that exposure isn't enough information for an attacker to access funds in a well-built multisig wallet. Additional keys are needed that the particular hardware wallet doesn't possess. Those additional keys could be created independently, with other hardware wallets from completely different manufacturers, and stored separately. This compartmentalizes and contains the risk of a vulnerability discovery, affording multisig wallet owners more time to react and replace the affected key or keys. The Coldcard incident demonstrated that Coldcard users in a multisig arrangement had far greater protection from the catastrophe.

If you'd like to learn more about setting up multisig protection, our team is here to help. We provide expert guidance, education, and support for navigating multisig bitcoin custody with confidence. 1 of every 200 bitcoin that will ever exist is secured with an Unchained vault. Multisig wallets are the foundation of all our products and bitcoin-focused financial services, from IRAs to inheritance. Learn more and ask questions on a free consultation call, and we hope to work with you soon.

Selected primary sources

Hardware-wallet manufacturers

1. Trezor. Message Workflows (seed-creation inputs); Secure Elements in Trezor Safe Devices; How Trezor Safe 7's Three Hardware Layers Protect Your Wallet; and Trezor Hardware Built-In Security. Available from Trezor documentation and the Trezor firmware repository.
2. Ledger. Master Seed; OS Random Number Generation; Nano X and Stax product specifications. Available from Ledger Developer Documentation, Ledger Donjon, and Ledger product documentation.
3. COLDCARD. Master Seed, Security Status, Physical Notes, Secure Elements, hardware bill of materials, and firmware release history. Available from COLDCARD documentation and the COLDCARD firmware repository.
4. BitBox. BitBox02 Security Features and Threat Model. Available from BitBox and the BitBox02 firmware repository.
5. Foundation. Passport Security Model; Passport Prime audit response and security audit; and public Passport hardware repositories. Available from the Passport firmware security documentation, Passport Prime audit response, and Passport electronics repository

Chipmakers and secure-element vendors

1. STMicroelectronics. STM32 data sheets, RNG application note AN4230, and STM32U5 product-release documentation. See AN4230.
2. Infineon. OPTIGA Trust M V3 Product Brief, including the stated Common Criteria platform and AIS 31 PTG.2 claims. Available here.
3. Microchip Technology. ATECC608-series and SAMA5D2 data sheets and product literature, including ATECC608C documentation. See the ATECC608C data sheet.
4. Analog Devices / Maxim Integrated. DS28C36B secure-authenticator documentation and reliability records. See DS28C36 documentation.
5. Espressif. ESP32 and ESP32-S3 hardware-randomness documentation. See ESP-IDF Random Number Generation.
6. Silicon Labs. EFR32MG24 data sheet and Secure Vault documentation. See the EFR32MG24 data sheet and Secure Vault.
7. Tropic Square. TROPIC01 public RTL, security reports, and certification disclosures. Available from the TROPIC01 repository, security-report index, and Tropic Square FAQ.

Selected primary sources

Certification and validation records

1. NIST Cryptographic Module Validation Program, Entropy Source Validation. Public validation records for STM32U5 implementations and Microchip ATECC608-series devices, including certificate E46 and the NIST entropy-validation search register.

2. Common Criteria portal. Public certification reports and security targets for ST secure-element families, including the ST33J2M0 report and ST33K1M5 report.

This article is provided for educational purposes only. Unchained does not represent that the techniques, software, and/or hardware referenced by this article are appropriate for your given use case or legal in your given jurisdiction. Choice of a hardware device is a serious matter. Do your own research before choosing a device and all use is at your own risk, as Unchained has not performed security review on the devices mentioned herein. For any questions related to your own specific situation, please consult with your own attorney, tax professional, and/or licensed financial advisor, and do your own research before using any specific technique, software, or hardware for purposes of storage or transmission of any bitcoin. Information has been obtained from sources that we believe reliable, but we do not warrant or guarantee the timeliness or accuracy of this information.