Xecret.io

Xecret App

www.xecret.io

Next-generation Non-custodial Cold-storage Application

Last updated - 11/2021

Overview

- This document describes a mobile and web application for physical cold storage (or backup) and recovery of sensitive and valuable information. The application protects confidential information such as credentials, master passwords, private keys, and recovery (seed) phrases simultaneously against loss, physical destruction, and theft.

- The application leverages a decentralized method to provide the most secure and reliable way to protect any string of text that has 2,000 characters (about 2kb) or less, offline.

- The application—Xecret.io—works by creating next-generation cryptographic data slices from the original secret information in the form of QR codes, called “Xecrets.”

(Visual Xecret Representations as a set)

For each item of secret information, users create a set of Xecrets, and no individual Xecret will contain enough data to recompile the original secret information. This renders the technology hack-proof. In other words, any entity with custody of one Xecret in a set will not have access to the secret information, either in whole or in part.

The Issues

Issue #1: Trust and Security

One of the more significant challenges in information security is that of trust. With secure systems, the goal is to eliminate the need for trust. Therefore, commodity cloud storage solutions protect the secrecy of file contents by applying end-to-end encryption. Blockchain and similar technologies deal with trust in transaction and ownership integrity by making violations so computationally difficult that they are practically impossible.

Similarly, the task of offline or cold storage of secret information, such as recovery passwords or seed phrases, must alleviate the issue of trust. Delegating the storage of a secret to a third party is not sufficient. That third party may betray the trust or become subject to a digital or physical hack, leading to a leaked secret.

This issue does not necessarily concern a lack of trustworthy parties. We assume most individuals have access to places or people such as colleagues, friends, or legal agents who are generally trustworthy. Rather, the problem is that no single person (or place) is trustworthy enough to be wholly responsible for protecting a secret against loss, physical destruction, or theft.

Issue #2: Inverse proportionality in Redundancy Levels

In today's digital environment, you must take every measure to prevent valuable information from loss, destruction, or theft. In a space like crypto, there is zero tolerance for such failures of information security with no recourse for retrieving funds. Security experts say that the best practice is to keep **one copy** of valuable information such as private keys offline in a safe and secure place. Offline storage, however, has its own challenges. For example: How can you guarantee that your single copy of secret information will not be compromised or destroyed by loss, theft, or natural disasters? To protect against disasters such as fire, security professionals recommend that you make **multiple copies** and distribute them among multiple locations to achieve reasonable redundancy. In the event one copy is lost, other copies remain available. But having more than one copy increases the chance of theft. This creates a dilemma with no ideal number of copies to store.

According to *The New York Times*[1]:

> *The Winklevosses (from Gemini Trust Co) came up with an elaborate system to store and secure their own private keys. They cut up printouts of their private keys into pieces and then distributed them in envelopes to safe deposit boxes around the country, so if one envelope were stolen the thief would not have the entire key.*

While we should applaud their method, the inherent issue here is that if even one of the envelopes gets lost, Gemini will lose its cold storage backup. Since a part of the key will be missing, the private key string can't be recompiled to the original. Worse, the thieves will have a piece of the private key that they didn't have before, which provides them an exponential advantage to launch a brute-force attack.

When storing secret information such as credentials, private keys, and recovery keys offline, there are two major hazards against which the secret needs uniform protection:

fire and theft.

Currently, no cold-storage solution can protect simultaneously against fire and theft. Various levels of redundancy exist in various solutions, causing two opposing vulnerabilities:

Specifically, when you protect against fire by distributing multiple copies, the protection against theft inevitably erodes. Inversely, when reducing the number of copies to protect against theft, the vulnerability to fire and theft increases. As shown in the equation below, the benefits of each type of protection are inversely proportional:

Protection against fire (X) = 1 / Protection against theft (Y)

The more redundancy (i.e., the greater the number of copies), the better protection achieved against natural disasters such as fire. Increased redundancy, however, increases the chance of theft.

Vulnerability Due to Increased Redundancy: Producing and distributing more copies of the secret provides better redundancy or less dependency on an individual copy. If a fire or other natural disaster destroys one of the copies, or if an entity such as a government authority denies access, the other copies remain available. Having several copies, however, will increase the owner's vulnerability; one or more of the keepers may betray the owner, or one or more of the copies could become otherwise compromised. Thus, each copy distributed to a location or person creates increased proportional risk. For example, if you distribute five copies, the risk to the security of the information increases by a factor of five.

Vulnerability Due to Decreased Redundancy: By decreasing redundancy, the chances of theft also decrease. Having fewer copies, however, also increases the chance of complete data loss if all become either inaccessible, lost, or destroyed.

Current solutions do not provide for comprehensive protection. As discussed above, there are opposing security issues to reconcile, and providing the best-available protection has been a double-edged sword:

Available current solutions do not provide for comprehensive protection. As discussed above, there are opposing security issues that need to be reconciled, and providing the ideal protection has been a double-edged sword:

> **When implementing additional security measures against one issue, we create a new set of security vulnerabilities against the other.**

The individuals must make a choice when it comes to offline storage: Which element carries a higher level of risk and tailor a solution that protects against either fire or theft.

Until now, offline storage has forced us to choose: Which element carries a higher level of risk? Only after making that choice can we tailor a solution that protects against either fire or theft.

We need a new cold storage solution to protect secret information against **loss, natural disasters, and theft equally and simultaneously,** without compromising any security

Solution

The ideal solution to this problem makes use of available resources in a balanced approach that provides a very high level of assurance of data secrecy without needing to delegate trust to any single individual (or location). Even better, the solution is to specify a threshold number greater than two individuals (or locations), who collectively hold (or store) the key to recover the secret. If the individuals do not have direct or close relationships with each other, such that they would be tempted to collude and recover the secret, their level of “general trustworthiness” is amply sufficient to give a very high degree of confidence in the overall secret—a degree far exceeding each party’s individual trustworthiness.

The solution involves a cryptographic algorithm that generates a set of secure slices of the original secret. But rather than slicing a secret like “A B C D E F” as “A B,” “C D,” “E F,” it instead uses a proprietary Private.me algorithm that slices as something more like “I U A S N M,” “Z X O E J U,” and “N W E Y G S” plus any number of additional slices. This is just a conceptual illustration of how each secure slice is the same length as the original secret, **yet completely impenetrable and opaque, so we can learn nothing about the original secret from a given slice.**

In reality, each slice would typically be a cryptographic QR code, a scannable representation of the binary data. You could print these QR codes and store them as physical media in secure locations. We call these cryptographic QR codes “Xecrets.”

The algorithm allows the user to specify: (1) the number of Xecrets in each set, and (2) the threshold number of Xecrets for the set, or the number of Xecrets necessary to recover the original encrypted information. This way, you can store the slices in distinct secure locations or distribute them to multiple reliable parties. The threshold slices make up a decentralized cold storage wallet. Later, when the owner wishes to recover the secret, they recover the threshold number of their Xecrets, which allows them to decrypt and recover the original secret offline (or online).

Because each Xecret is unhackable, the application provides the highest level of security and privacy that we know of. Once the user sets the threshold number for a set of Xecrets, they can use any combination of Xecrets from the set to recover their original secret information as long as that combination includes the threshold number (or more) of Xecrets. This allows the application to facilitate the highest level of redundancy, availability, and accessibility with minimum dependency.

No individual Xecret from a set has enough information to decrypt the original information—even using the most powerful computers, including future quantum computers. This renders each Xecret unhackable when separated from the other Xecrets in its set.

The number of Xecrets in a set and its recovery threshold are flexible. Users can increase either or both to enhance security and redundancy (for example, a set of five with a threshold of three, a set of nine with a threshold of four, etc. depending on their needs).

Due to the decentralized state of the secret information, each individual or location holding a Xecret, or "Xecret Keeper^{TM}," will not have control over or custody over the original information. This drastically reduces Xecret Keepers' risk in their custodial responsibilities. If one of the Xecrets is lost, stolen, or destroyed, the application can still recover the original information from a threshold number of other Xecrets in its set.

The information stored within each Xecret is in a decentralized structure as opposed to a unified structure. It therefore has some inherent protections similar to those that blockchain provides:

- Blockchain technology requires multiple nodes to validate a transaction. Similarly, Xecret.io needs multiple Xecrets to decipher the original encrypted information.
- Also like blockchain technology, a successful attack on one node or one Xecret is not effective. Blockchains are designed so that 51 percent of the nodes must be attacked successfully for the attack to be effective. With Xecret.io, users choose a threshold level to mitigate known and unknown risks.

However, Unlike blockchain technology, each Xecret stores only a part of the randomized data-set, whereas blockchain records the whole data-set many times over.

Technology

We built the application as a standalone web app and recommend that you use it offline. All processing occurs on the client side, as remote servers are not trusted with the user's secrets.

The Xecret.io application uses a multilayer protection method to secure any string of private information. First, the application encrypts the secret with 256-bit AES[2], then it applies All or Nothing Transform[3] (AONT) to its output and lastly uses a licensed Information Dispersal Algorithm (IDA) from Private.me to cryptographically slice the secret. This algorithm uses a bitwise XORing approach much like the common One Time Pad (OTP) algorithm[4], differing only by the need to spread the encrypted information among dispersed parties and to have the threshold number of Xecret Keepers to reassemble the original information.

The injected entropy from the IDA makes each Xecret computationally indistinguishable from random noise. A proprietary algorithm first injects the noise into each slice and then later—when the user wants to decrypt the encrypted information and has recovered the threshold number of Xecrets—separates the signal from the noise. If the user holds fewer Xecrets than the threshold during recovery, the data remains impenetrable, even given hypothetically infinite computing resources, just like an OTP.

Note: Given high-quality randomness of the original encryption and entropy injection, any number of Xecrets fewer than the specified threshold are immune to brute-force attacks. The only way to infer any details about the original information would have to be from incidental attributes, such as guessing the contents based on the length. We address this issue in "Risk and Mitigation," below.

The Science: Hack Proof Information Dispersal Algorithm (IDA)

The Private.me IDA algorithm[5] sits at the core of the security and privacy protections that the Xecret.io app provides. By guaranteeing that you must have a threshold of Xecrets to recover the encrypted confidential information, and dispersing those Xecrets to secure places, the system can defend against data compromise if an attacker breaks into a single Xecret.

The IDA works by introducing entropy or randomness into a Xecret, which prevents the brute-forcing of the encryption. It uses the same technique as the OTP algorithm, and as such is a theoretically secure encryption algorithm[6]. Given appropriate randomness, this renders **any brute-forcing of data encrypted under the IDA impossible**. Even if an attacker had infinite computing capacity and time, the encrypted data is indistinguishable from pure randomness due to the injected entropy.

Theoretic information security as described above reflects a **level of assurance higher than merely algorithmic security like the AES cipher uses**. In the case of AES, a comparatively short key drives a complex algorithm to obscure the resulting ciphertext. Given sufficient computing resources, hackers can break an AES cipher, unlike the Private.me IDA or the OTP algorithms.

The Xecret.io algorithm runs locally on the user's machine, not on a remote server. Some software service providers avoid sharing their code to protect their trade secrets, however we believe that local execution is an important part of the Xecret system's trustworthiness. As discussed above, the very idea of placing "trust" in third party systems should be avoided, as spreading trust too widely increases attack surface area. By running code locally, users are assured that there is no "man in the middle" attack nor any opportunity for compromise of their most sensitive secret information. Furthermore, the locally running code could even be **inspected and audited** to assure a user of its safety and correctness. This local execution arrangement ensures maximum information security assurance and minimizes potential attack surface area.

Cryptographic Process of Exclusive-Or (XOR) Encryption

Exclusive Or (XOR) is a simple process and very efficient computationally. The logic compares two value bits and generates one output bit. If the bits are equal, then the output is 0. If the bits are different from each other, then the output is 1:

XOR Logic:

If 0 and 0	then	Same Bits	Thus; 0 XOR 0 = 0
If 1 and 1	then	Same Bits	Thus; 1 XOR 1 = 0
If 0 and 1	then	Different Bits	Thus; 0 XOR 1 = 1
If 1 and 0	then	Different Bits	Thus; 1 XOR 0 = 1

Encryption Process:

Secret Text	10011001
	XOR
Random Bytes	01010101
Cipher Text	11001100

Decryption Process:

Cipher Text	11001100
	XOR
Random Bytes	01010101
Secret Text	10011001

Exclusive-Or Information Dispersal Algorithm (XOR IDA)

Data security experts categorize IDA algorithms specifically designed to address the security and privacy of dispersed data as "secret sharing" or "secret splitting" algorithms[7]. The Private.me IDA fits this category.

One of the simplest secure secret-sharing algorithms is the OTP. Given appropriate padding and a sufficiently long random key, the OTP compares favorably with the best encryption algorithms due to its attributes of perfect secrecy and ideal memory constraints (see the following discussion for an explanation of these terms).

An OTP can be thought of as a two-of-two IDA, in that you must have two parts out of two total dispersed shares to reassemble the plaintext. In this case, the random byte string is one of the secret shares, and the (secret XOR randoms) is the other share. If an attacker possesses the random byte string, they obviously have no information about the secret. And similarly, any data that has been XORed with random bytes becomes indistinguishable from random bytes, so the other share is secure as it stands alone as well. Reversing the OTP is as simple as XORing the two secure shares together, effectively factoring out the entropy of the randomness and restoring the original encrypted information.

As illustrated in Figure 1 below, XOR is a bitwise mathematical operator that performs an eXclusive OR, such that it sets the output bit if and only if exactly one of the two input bits is set. If both input bits are set, or if both input bits are cleared, then the XOR output bit is cleared.

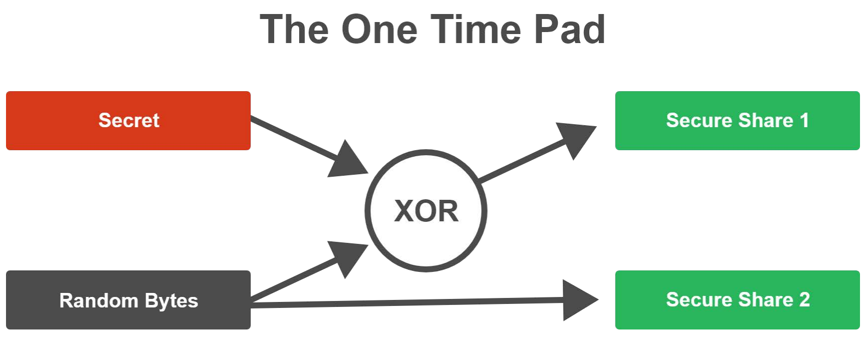

Figure 1

Every binary bit of an output slice or share is XORed with a binary bit of randomness or entropy that is used only once within that slice. The formula f () may be simple or complex. In this case, it is complex due to extrapolating the OTP to work within a threshold of multiple shares. But the fundamental security that results from XORing with unique entropy remains the same. Figure 2 illustrates the IDA entropy injection formula.

IDA Entropy Injection Formula:

$$S_n = f(d_n) \otimes r_n$$

n: bit index
s: slice
f(): transformation function
d: plaintext
r: randomness/entropy
$\otimes$: bitwise XOR (i.e. eXclusive OR)

Figure 2

Private.me Exclusive-Or Information Dispersal Algorithm (XOR IDA)

The XOR IDA as applied in the above overall data dispersal involves a straightforward extension of the OTP algorithm described above, changing it from a two-of-two IDA to a two-of-three IDA. As such, two shares are required to recover the encrypted information out of three total shares dispersed to the cloud. In other words, any two of the three shares a.k.a. Xecrets are sufficient to recover the encrypted information. Higher levels of thresholds are also possible, such as three-of-five or four-of eleven schemes. We are illustrating the two-of-three dispersal scheme here, however, for the sake of simplicity and clarity.

As Figure 3 shows, you can think of each Xecret as a sequence of bytes, wherein each byte is a sequence of bits. Each bit of the output is allocated a unique, random bit from the random number source. If each resulting bit of the share is either a random bit or a random bit XORed with a data bit, the property of perfect secrecy remains intact.

Private.me XOR IDA: Dispersal

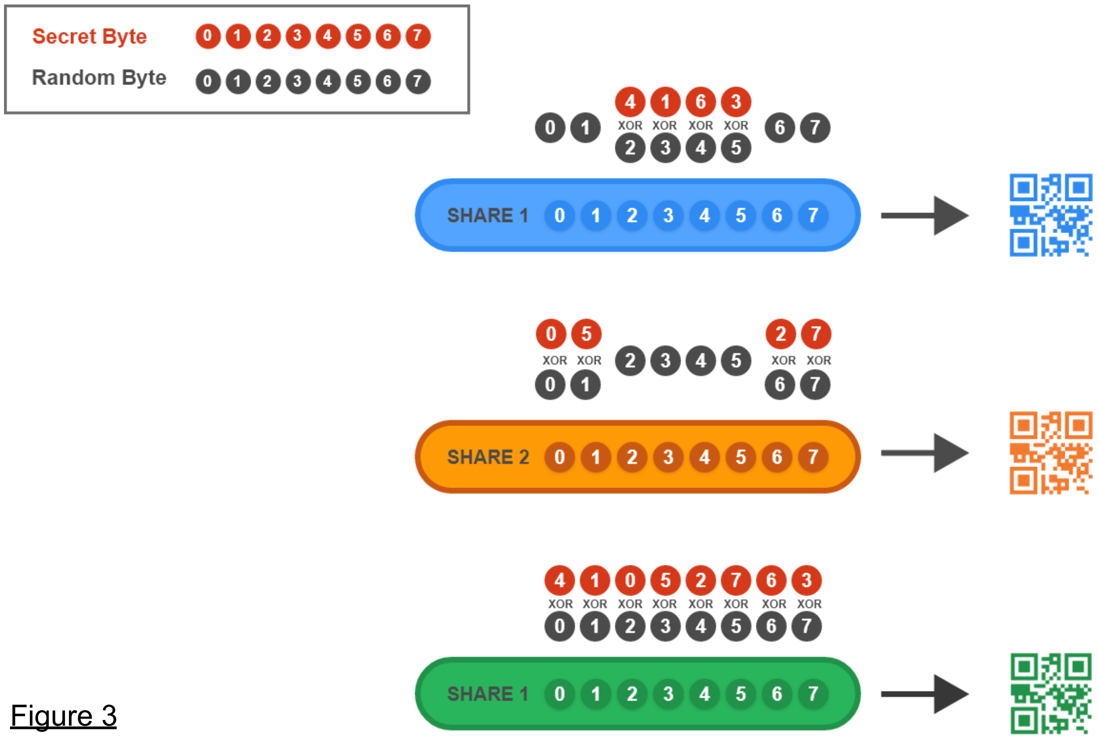
Secret Byte 0 1 2 3 4 5 6 7
Random Byte 0 1 2 3 4 5 6 7
0 1 | 4 XOR 2 | 1 XOR 3 | 6 XOR 4 | 3 XOR 5 | 6 7
SHARE 1 0 1 2 3 4 5 6 7
0 XOR 0 | 5 XOR 1 | 2 3 4 5 | 2 XOR 6 | 7 XOR 7
SHARE 2 0 1 2 3 4 5 6 7
4 XOR 0 | 1 XOR 1 | 0 XOR 2 | 5 XOR 3 | 2 XOR 4 | 7 XOR 5 | 6 XOR 6 | 3 XOR 7
SHARE 1 0 1 2 3 4 5 6 7

Figure 3

Private.me XOR IDA Reassembly

Reassembly of the dispersed Xecrets depends on which shares are available. The procedure is different if, for example, you have Xecrets one and two versus Xecrets two and three. Thus, there are three reassembly algorithms when two Xecrets are involved. Figures 4, 5, and 6 depict these operations.

XOR IDA: Reassembly of Shares 1 and 2

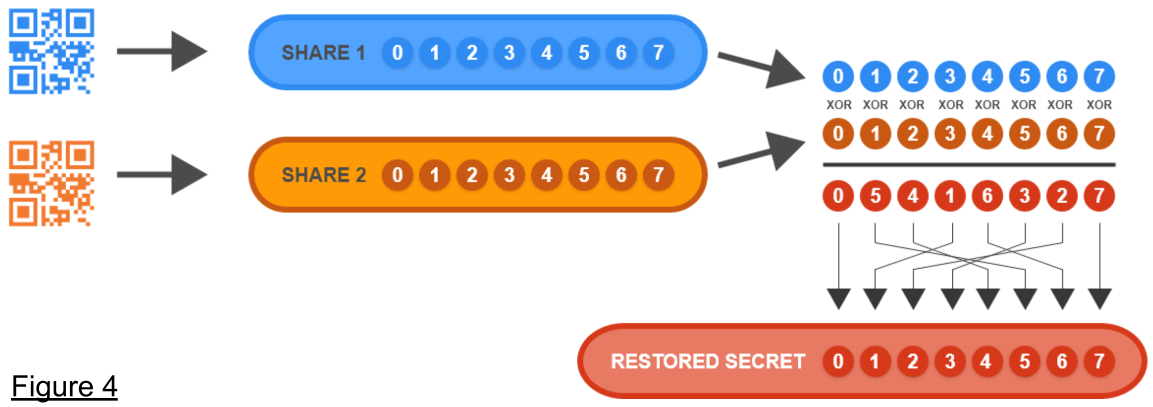

Figure 4

XOR IDA: Reassembly of Shares 1 and 3

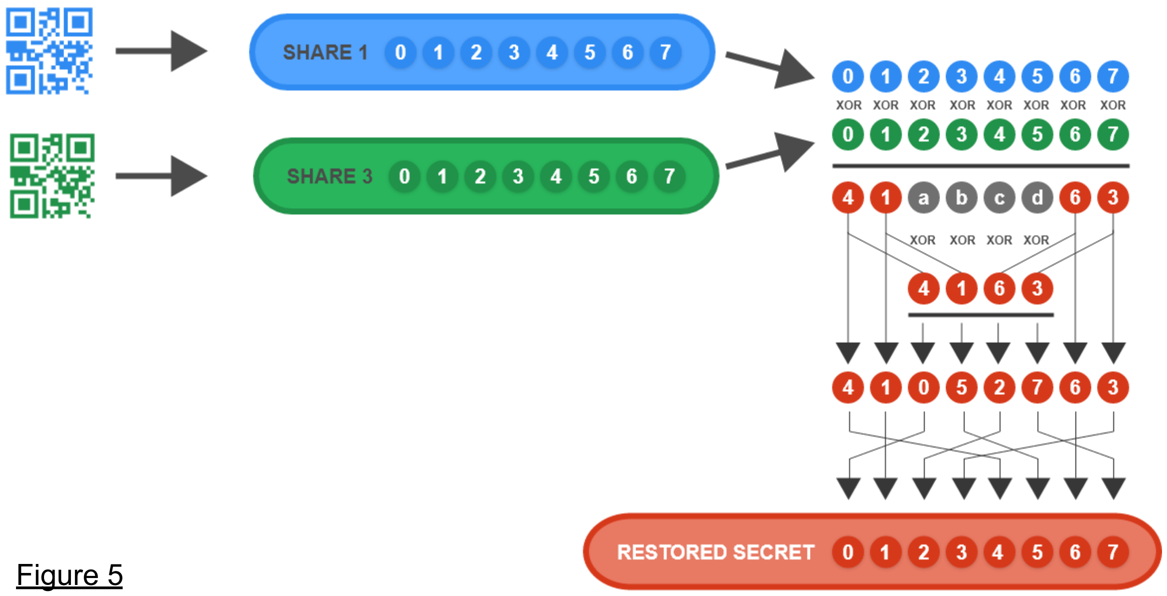

Figure 5

XOR IDA: Reassembly of Shares 2 and 3

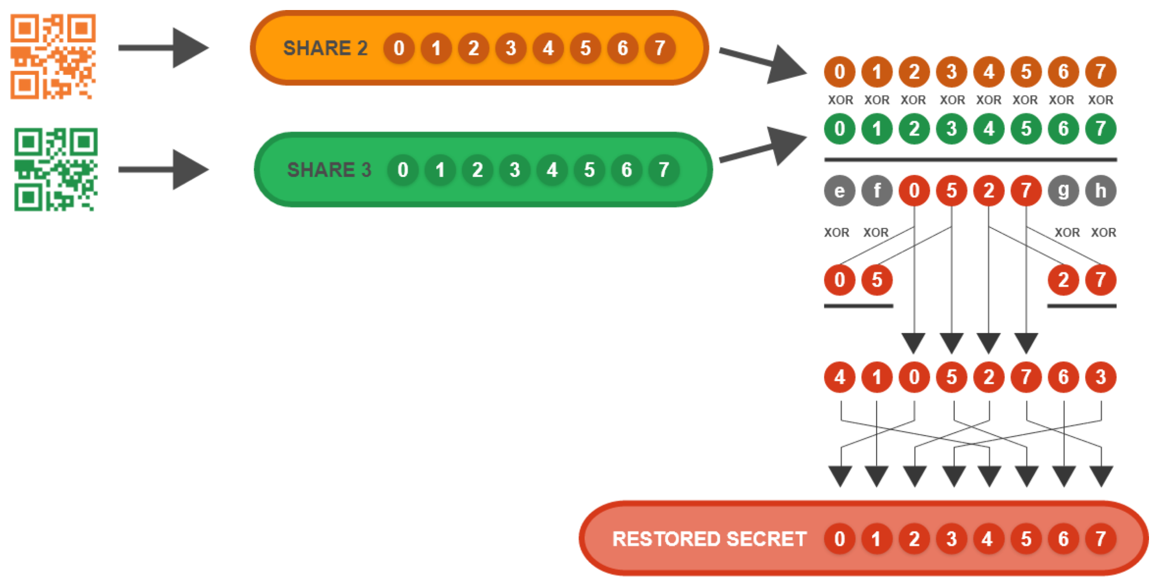

Figure 6

Note: letters a, b, c…, f, g, h above represent intermediate products of the IDA reassembly stage. Their value is only important for purposes of bit algebra through the dispersal and reassembly logic to reach the desired results. For example, by inspecting the dispersal step, the circled "a" above must represent (secret byte index 4) xor (secret byte index 0).

Private.me Patents:

Secure distributed data storage

Issued	Jun 21, 2016	9,374,422 B2

Secure and private data storage

Issued	Jun 2, 2020	10,671,760

Information Dispersal for Secure Data Storage

Pending	Aug 6, 2021	17/395,661

Conclusion

Despite the amazing strides in encryption technology in recent decades, a dilemma lurks at the core of the quest to achieve perfect security for sensitive confidential information.

Making multiple copies of the information reduces the risk of accidental or deliberate destruction if those copies are stored in environments with disparate vulnerabilities (one location might be at higher risk of destruction by hurricane flooding but at lower risk from destruction by fire, while another location might have the reverse risk profile). But creating multiple copies of sensitive confidential information increases the risk of loss by theft proportionally. The dilemma results in an x = 1/y situation, in which reducing the risk to the security of the information by one strategy (increasing the number of copies of it) increases the risk posed by the only other strategy currently available (limiting the number of copies) and vice versa.

Xecret.io has developed technology that not only escapes this dilemma but also combines the strengths of the two traditional strategies. It uses proven and proprietary algorithms to dissect the original confidential information into unhackable encrypted portions called Xecrets. Users define the number of Xecrets in a set and the number of Xecrets from a set that is necessary to recover the original information. It thus combines the advantages of single-copy storage (only the user needs to know how many Xecrets encode their information and how many Xecrets are necessary to recover the information) and multiple-copy storage (Xecrets can be dispersed to multiple locations with disparate risk profiles), while eliminating the risks of both.

References

[1] https://www.nytimes.com/2017/12/19/technology/bitcoin-winklevoss-twins.html

[2] https://en.wikipedia.org/wiki/Advanced_Encryption_Standard

[3] https://en.wikipedia.org/wiki/All-or-nothing_transform

[4] https://en.wikipedia.org/wiki/One-time_pad

[5]https://en.wikipedia.org/wiki/Private.Me

[6] https://en.wikipedia.org/wiki/Information-theoretic_security

[7] https://en.wikipedia.org/wiki/Secret_sharing

Risks and Mitigations

Risk 1: An attacker may guess the contents of a single Xecret based on its length.

Mitigation: There will be a maximum size of information that can be encrypted in a set of Xecrets, based on the scannable codes involved. For any private information smaller than that size, Xecret.io will pad each Xecret to the maximum length. Thus, the length of an individual Xecret will reveal nothing about the information it encodes.

Risk 2: A person may fail to store a Xecret securely or may lose it.

Mitigation: The application includes downloadable, standardized info sheets that users can share with their Xecret Keepers when negotiating their agreement to store the confidential information. The info sheet could include acceptable approaches for both hiding and securing the Xecret.

Risk 3: A threshold number of Xecret Keepers may collaborate and betray the user who created their Xecrets.

Mitigation: When sharing with people, users should choose people who don’t know each other.

Risk 4: Someone might create a public registry of the names of Xecret.io users, which might encourage unscrupulous Xecret Keepers to register the name of the user who entrusted them with their Xecrets, look for other Xecret Keepers in service to the same user, and collaborate to recover the confidential information.

Mitigation: Ultimately, this system rests on the trustworthiness of the Xecret Keepers, be they individuals or corporate entities. Each Xecret.io user should choose their Xecret Keepers carefully. The extremely risk-averse, who must protect a critical item of confidential information, can use safe deposit boxes at different banks, taking humans almost entirely out of the equation.

Risk 5: Photos taken on practically any device that connects to the Internet can propagate those photos automatically to iCloud Photos, Google Photos, or other cloud storage options. If a full set of Xecrets existed on one phone, the dispersal element of Xecret.io would collapse and allow a cloud administrator, or perhaps a hacker, to recover the confidential information.

Mitigation: Literature included in Xecret.io instructs users to avoid adding photos of Xecrets to their camera roll or to immediately delete any such photos in their camera roll.

Risk 6: Scanning a Xecret may trigger the analytics incorporated into Apple or Google phones to silently send that data to a data warehouse in the cloud, violating secrecy expectations.

Mitigation: Xecret.io will offer a proprietary format for the physical Xecret, a format that consumer devices won't automatically read.

Risk 7: Web apps are less secure and trustworthy than those distributed through app stores. For example, even though the web app would be set up to do as much as possible client-side, and avoid sending secrets to the server, there are still risks.

Mitigation: Xecret.io instructs users to cut off internet access while processing Xecrets for either dispersal or retrieval. It further enhances the security of users' confidential information by instructing them to close all browsers and clear all browser history before re-enabling internet access. In many cases, this would protect a user from the consequences of a hacked and compromised application.

www.xecret.io