Whitepaper

Viral Encrypted Security (VES)

A solution to the current technological compromise between the full privacy of end-to-end encryption and the ability to recover lost encrypted content in the event the owner loses all copies of keys in their possession.

Authors: Jim Zubov and John Brixius

Version Date: 7 August 2017

Abstract

Current encryption technology is impractical for mainstream use on any stored content because of the tradeoff between full privacy and the safety of not losing information due to key loss. Viral Encrypted Security (VES) is a solution to this problem, offering the full privacy of end-to-end (e2e) encryption while also providing an easy, reliable and efficient means to recover encrypted content in the event the owner loses all copies of his/her keys.

VES uses industry standard AES and RSA encryption in conjunction with a scrambling process based on linear algebra, similar to Shamir's Secret Sharing.

Each user has a Shadow Vault that is encrypted by a distinct asymmetric key pair, different from the user's primary vault asymmetric key pair. The User pre-selects a number of friends, *N*, who can assist the User in Recovery of his/her encrypted content should the User lose all copies of his/her main passphrase, the VESkey. The User also selects the number of friends, *X*, needed to achieve Recovery. Using the scrambling process, the Recovery Key is translated into *N* Tokens, each representing a linear equation with *X* variables, whereby any X of the *N* Tokens are needed to solve the linear equations and reconstruct the Recovery Key. Any number of Tokens less than *X* is useless in reconstructing the Recovery Key – Tokens are essentially as difficult to unscramble as is 256-bit AES encryption.

Each Token is allocated to the vault of a friend, and is hence encrypted with each friend's public asymmetric key. Only that particular friend can unlock the Token. When the User loses his/her primary key, he/she creates a new key and all friends are notified. Each friend must enter his/her VESkey to unlock the Token, encrypt it with the User's new public key and send it to the User. When the User has received *X* Tokens, he/she can descramble the Recovery Key and use it to unlock the Shadow Vault to retrieve the lost contents, which are promptly re-encrypted with the new primary vault key and deposited in the new primary vault. All old shadow vault items, keys and Tokens are then deleted.

The VES viral network is structured to enable an interactive chain reaction Recovery process. Once any user has Recovered their Shadow Vault, they have also Recovered any Tokens that were stored in it, and can then assist anyone who needs Recovery assistance, and so on. Every link in this chain reaction requires a user supplied, generally manually entered, VESkey to continue the chain reaction, making it essentially impossible for the chain reaction to continue from one person's vault to the next without the manual involvement of each person at each link in the chain. One user can simultaneously launch multiple chain reactions.

Using the below recurrent formula, it can be shown that even a small VES viral network of friends will realistically provide a very high probability of Recovery.

$$p_{L+1} = p_0 \sum_{i=0}^{x-1} \left[\left(\frac{N!}{(N-i)!\, i!} \right) p_L^{(N-i)} (1 - p_L)^i \right]$$

Variable *X* and *N* have already been identified. p_0 is the probability that any single user will lose his/her keys and *L* is the total Level of friends in the network (e.g., $L = 0$ is just the user, $L = 1$ adds the User's friends, $L = 2$ adds the friends' friends, etc.). For the situation of $p_0 = 30\%$, the User has 5 friends who each have five friends so that $N = 5$, $X = 2$, and $L = 2$ for a total of 31 people in the network, the odds that the User will not Recover the lost content are approximately 1 in 92 million. When the network extends to $L = 3$ the odds go to 1 in $4.8*10^{31}$. A properly setup VES viral network is a very effective and dependable backup for key loss.

The VES viral network also creates a unique 3rd factor authentication, independent from the traditional factors: *something I know* and *something I have.* Each friend can be required to voice or video verify the person seeking Recovery is actually the User they know before assisting in the Recovery process. Unlike information that can be stolen and stored for later use, such as a password or fingerprint, VES 3rd factor authentication cannot be conceptually stolen or stored. The VES 3rd factor also addresses the account theft vulnerability by blocking the thief from executing a Recovery as a means to bypass the existing VESkey. Generally, VES is as secure as non-VES e2e encryption for stored data.

VES Concept

The objective for Viral Encrypted Security (VES) is to maintain complete user privacy of the encrypted content but also provide a robust mechanism to recover the encrypted content (Recovery) should a user lose their personal, secret encryption key (VESkey). VES achieves this through a virally connected network of friends who can use their own VESkeys to assist the User who has lost his/her VESkey, but cannot decrypt the User's content or access it in any way.

While the odds of any single user losing his/her VESkey may be unacceptably high, this individual risk can be diversified away in a viral network of users. VES is designed so that only a small number of users need to remember their individual VESKeys so that the rest of the connected users in the network can recover their lost content.

An analogy would be a village of persons, each holding a torch on a somewhat windy day. The odds that any single torch gets extinguished at any point in time may be unacceptably high if any single torchbearer were alone. But the odds that *all* the torches go out simultaneously are extremely low. As torches become extinguished, the remaining lit torches are used to relight them. With a large enough population, and assuming it takes minimal time to light another torch (cycle time), the odds that a torch will not be relit approaches zero percent.

Concept Validation

A mathematical model has been developed to explore the dependencies and validate the extent of the applicability of the VES concept.

For simplicity, assume a network where each person has N friends who can enable Recovery of encrypted content when each person loses his/her VESkey. Each person has the same probability, $p_{0,}$ of losing his/her VESkey within a given span of time, and each person's Recovery configuration requires assistance of X friends to enable Recovery.

It is also assumed that the cycle time of a friend assisting in Recovery is small compared to the cycle time of users losing their personal VESkeys. Under this assumption, a chain reaction of assistance by friends can occur under a relatively static condition of lost keys within the network, and additional key loss events are generally not occurring while the Recovery process for the first lost key is still in process. Assuming the Friend has access to a connected device, the time it takes to assist in Recovery ranges from a few seconds to a few minutes (for the voice verification). With an *N* sufficiently larger than *X*, it is reasonable to assume that significantly more than *X* of the *N* friends will be logged into their computers, or have email alerts enabled on their cell phones, and can immediately offer assistance.

Under these assumptions, viral, networked Recovery can be modeled by the following recurrent formula:

$$p_{L+1} = p_0 \sum_{i=0}^{x-1} \left[\left(\frac{N!}{(N-i)!\, i!} \right) p_L^{(N-i)} (1 - p_L)^i \right]$$

where *L* is the level of the network depth.

Starting with $L = 0$, which represents the lone User (without yet including his/her friends), $p_L = p_0$. p_0 is then used to calculate p_1, the probability of losing data when relying on assistance from only a single Level of *N* friends, $L = 1$, and not yet including any friends in subsequent Levels (e.g., $L = 2$ and beyond).

With p_1, p_2 can then be calculated for the case when each of *N* friends has his/her own *N* friends to assist with Recovery, and so on.

For an estimate, let's start with the assumption that the average probability that any person loses their VESkey is 30% over the course of an assumed average cycle time between logins of one week. (Users will discover their key is lost when logging in, so the cycle time for which keys are lost will be the average time between logins.) Note that this assumes all VESkeys are manually entered by users and not stored on their devices. If stored on a device, the probability of losing the VESkey will be much lower than 30%.

Also, assume that $N = 5$, or the primary User has five direct friends who each have five direct friends and so on. Lastly, assume that each person requires the response of 2 of their 5 friends to achieve Recovery, resulting in $X = 2$.

To keep the example simple and conservative, assume that there are only two levels of friends in the network ($L = 2$). Also, it is assumed that each person is only used once as a friend, and that the network is closed with no additional friends beyond these levels (both are unlikely real world condition). This results in a total of 31 distinct people in the network including the original User. In this case, the odds that the original User permanently loses his/her content from key loss is 1 in 92 million.

If this example were extended to a third level of friends, $L = 3$, for a total of 156 people in the network, the odds grow to 1 in $4.8*10^{31}$, which is effectively zero percent. To experiment with the formula for various scenarios, go to: https://www.vesvault.com/fun-math.

Also, adding Levels of friends beyond three levels essentially mitigates the possibility of not Recovering content even if the probability that any single person loses their VESkey is unreasonably high. For example, with a $p_0 = 75\%$ and with a network that extends to $L = 4$ for 781 total people, the odds of no Recovery are approximately 1 in 1.4 trillion. Thus, the depth of the network essentially compensates for any realistic amount of individual human risk, p_0, of VESkey loss. Again, this is assuming the worst-case scenario for p_0, that all users manually enter their VESkeys and do not store them in their local devices. The p_0 would be expected to be much lower if all or part of the VESkeys were stored in the users' devices.

Two considerations of the model should be noted. First, the model implies that each person in the network is unique and only used once as a friend. In reality this condition will not hold as individuals will be selected as friends multiple times with persons anywhere in any network, also creating the condition where two people can assist each other. However, if N is estimated to be the number of *unique* friends at each level and not the total number of friends, the model conservatively

addresses this condition and will result in an estimated probability of no Recovery that is somewhat higher than a more realistic estimate. For example, since the median number of friends for a user of facebook is 200, our example of $N = 5$ unique friends is very conservative for this sort of viral network. This is especially conservative in that only two levels of friends are used in our example, where in reality it is quite likely that the network will extend beyond $L = 2$.

It is important to note that multiple overlapping connections of friends do not diminish the probability of Recovery for a given sized network. In fact, *if* Distributed Recovery is being deployed (i.e., more than one friend is required for Recovery), more direct first level friends in the network actually *improve* the odds of Recovery. For example, in our example of the network of 31 total persons and Distributed Recovery with $X = 2$, all 31 persons can select each other as first Level friends (i.e., $N = 30$ for any one user) and have no second Level. In this alternate arrangement of the 31 people, $N = 30$, $X = 2$ and $L = 1$, and the probability of no Recovery will actually *decrease* compared to the $N = 5$ example for the same 31 people, from 1 in 92 million to 1 in 228 trillion.

Second, having multiple connections means that there are more humans who can help with Recovery. Since the shortest overall cycle time will result in Recovery, more possibilities accelerate the chain reaction of Recovery throughout the network. As previously inferred, the faster the cycle time of Recovery compared to the cycle time of key loss, the greater the increase in real world probability of Recovery and the more effective and robust the Recovery process.

With all these considerations in mind, a properly setup VES Recovery network will provide an acceptably low risk of loss of data from key loss.

VES Applied Math

The concept of Distributed Recovery within VES, or that more than one friend is necessary to assist the User, is part of the requirement that no friend, or the

service provider, ever possesses a duplicate key or can access a key that can be used to decrypt the User's content. Instead, each friend possesses a Token, or scrambled version of the User's Recovery Key. To ensure that VES offers no less privacy and security than encryption without VES, the Token must be as difficult to unscramble as it is to brute force decrypt the encrypted content. In this regard, no friend, hacker or even administrators of VESvault, who comes in contact with a Token has a means of accessing the encrypted content with any more probability than brute force hacking the encrypted content.

A second, less critical, requirement of Distributed Recovery is that each friend can assist in Recovery independently of whether or not any other friend has also assisted in Recovery. This is an important enabler to improving the odds of Recovery for a given size VES network, but is not a requirement for a viral network to exist.

In summary, any combination of a predefined number of the friends' Tokens should allow for Recovery by reconstructing the Recovery Key, while any lesser set of Tokens is useless in Recovery or accessing the encrypted content.

Mathematically, this translates to N being the number of distinct Tokens $T_1 \ldots T_N$, and X being the minimum number of Tokens needed to reconstruct the Recovery Key. Any subset of X Tokens from $T_1 \ldots T_N$ is sufficient to recreate the Recovery Key while any smaller subset cannot do so.

This formulation suggests of a system of linear equations with X independent variables $V_1 \ldots V_X$:

$$C_{1,1}V_1 + C_{1,2}V_2 + \ldots + C_{1,X}V_X = T_1$$
$$C_{2,1}V_1 + C_{2,2}V_2 + \ldots + C_{2,X}V_X = T_2$$
$$\ldots$$
$$C_{N,1}V_1 + C_{N,2}V_2 + \ldots + C_{N,X}V_X = T_N$$

As long as there are X independent linear equations, the system is unambiguously solvable.

Since the variables represent binary data, loss of numerical precision is not acceptable. To ensure this doesn't happen, signed integers of unlimited length are used.

As for the coefficients $C_{i,j}$, the following considerations apply:

- The system has to be independent in order to be solvable, which means the determinant of the matrix $|C|$ must not equal 0.
- Zero values for any of $C_{i,j}$ may facilitate finding certain variables V_i from an incomplete set of equations, and should be avoided.
- Values of $C_{i,j}$ ending with sequence of binary 0's may reduce the entropy of the data, and should also be avoided.

As for the first provision, a known example of a matrix with a guaranteed nonzero determinant is the Vandermonde matrix $C_{i,j} = (b_i)^{j-1}$, as long as all bases b_i are distinct.

Using the Vandermonde matrix results in mathematical equivalence to Shamir's Secret Sharing algorithm, but described from a linear algebra point of view while Shamir defines his method from the polynomial perspective. An alternate, non-polynomial based matrix for the coefficients would make the algorithm mathematically different from Shamir's Secret Sharing.

A base value of 0 has to be avoided, as it would produce a trivial insecure equation $V_1 = T_1$.

Even numbers, particularly multiples of 4, will produce coefficients with trailing binary 0's, and for the reason mentioned above, should be avoided.

Negative bases, in case of signed operations, do not seem to provide any particular advantage and therefore are not used.

Any subset of the Vandermonde matrix with positive bases, in ascending sequence, is guaranteed to have a positive nonzero determinant, due to properties of Schur polynomials. This provides an extra level of protection from partially solving an incomplete set of equations.

Large bases will produce excessively large values for Tokens, in the case of using unlimited length integers, and should be avoided.

With all that said, a natural sequence 1, 2, 3, 5, 6, 7, 9, … (avoiding multiples of 4) reasonably meets the requirements for the coefficients.

Moving on to the desired requirements for variables $V_1 \ldots V_X$:

- All variables should be of the same order of magnitude, to avoid a possible degradation of entropy.
- As an extra security measure, only the complete set of $V_1 \ldots V_X$ should make it possible to reconstruct the Recovery Key, R. A partial set of variables should not reveal any useful part of R, nor significantly reduce the entropy.

One possible approach to generating the variables is as follows:

- Generate an intermediate vector $U_1 \ldots U_X$, assign $U_1 = R$, and U_2 through U_X are random or pseudorandom values of the same order of magnitude as R.
- Set V_i to a symmetrically encrypted value of U_i using U_{i+1} as a key, with the last value being set as $V_X = U_X$.

Having the complete set $V_1 \ldots V_X$, where the values $U_1 \ldots U_X$ can be decrypted in the opposite direction, will result in descrambling the Recovery Key, $R = U_1$. Without having the complete set $V_1 \ldots V_X$, the chain of decryption is not possible.

In the degenerate case of $X = 1$, the procedures described above will produce Tokens that are the same as the Recovery Key without any scrambling, $T_i = R$. Thus, for maximum security, X should always be greater than or equal to 2.

The size of each of the Tokens, T_i, produced using the method described above, is equal to or somewhat greater than the size of the Recovery Key, R. This size is not a concern for small 256-bit Recovery Keys, as currently used in VESvault. However, if the Recovery Key becomes significantly large due to adoption of

different algorithms in the future, the size of the Tokens may become a concern. In this case, a few extra steps can be taken:

- The intermediate values U_1 … U_{X-1} can be assigned approximately equal slices of R, while the last one, U_X, still being random. This will reduce the size of the Tokens, especially when X is large, without unduly compromising the security.
- Instead of using unlimited sized integers, all operations can be done on a modular field with a sufficiently large prime modulus M. This will produce Tokens of a fixed size with an acceptable length. The system is solvable unless the determinant of the coefficient matrix equals 0 modulo M, which is negligibly unlikely for a large M.

Process Implementation

VESvault uses industry standard 2048 bit RSA asymmetric encryption and 256 bit AES symmetric encryption methodologies, with the possibility of easily adopting alternative algorithms should the need arise. Also, a 256 bit scrambling process is used in conjunction with the encryption, resulting in a level of entropy on par with 256-bit AES encryption. All process steps assume these standards.

Except for content that is small is size, such as passwords and notes, all user submitted content intended to be kept secret is encrypted and stored separately from the structure outlined in this section. This section pertains to the repository of the keys that are used to encrypt and decrypt this content.

Vault Items are the metadata for Vault Entries. Each Vault Entry points to one and only one Vault Item but a Vault Item can have multiple Vault Entries pointing to it. Being metadata, Vault Items are not encrypted.

Vault Entries are encrypted data. Currently there five types of data stored as Vault Entries: (1) Keys, which are symmetric encryption keys for user submitted content; (2) Recovery Tokens; (3) user submitted passwords; (4) user submitted small text

strings, and (5) secondary passphrases, either randomly generated or user supplied, pertaining to secondary Vault Keys for affiliated apps.

Each Vault Entry is encrypted by, and points to, a specific asymmetric Vault Key, where the public key is stored as open text, and the private key as cipher text having been encrypted by a separate symmetric passphrase. These symmetric passphrases are one of: VESkeys, randomly generated Recovery Keys, or secondary passphrases for affiliated apps.

VESkeys are either user generated, or randomly generated temporary VESkeys that are assigned to users who have not yet created their personal VESkeys. Temporary VESkeys enable users to share content and select friends for Recovery prior to the friend having set up his/her VESkey.

As data is deposited, the corresponding list of users is passed, identifying the list of Vault Keys to be used to encrypt the data. If a user does not have a VESkey, this is when the randomly generated temporary VESkey is created. Each of the resulting cipher text instances of the data is deposited as a Vault Entry, pointing to the corresponding Vault Item and Vault Key.

When Recovery is properly set up, for every Vault Entry there is a matching Shadow Vault Entry, both of which point to the same Vault Item. The Vault Entry and sister Shadow Vault Entry are different cipher text versions of the same source open text entry. Whereas the Vault Key encrypts the Vault Entry, the Shadow Vault Key encrypts the matching Shadow Vault Entry.

Shadow Vault Entries are used exclusively for the process of Recovery: when the User loses his/her VESkey and relies on his/her pre-selected friends to enter their own VESkeys to assist in Recovery of the lost information.

The Shadow Vault Key – the private-public asymmetric pair – is randomly generated when the User sets up Recovery. The Shadow Vault private key is then symmetrically encrypted using a randomly generated symmetric Recovery Key, *R*. After which, *R* is passed through the scrambling algorithm described in the

previous section to produce Recovery Tokens T_1 … T_N, that are deposited into each friends' vault.

The Vault Item for each Token consists of a reference to the User's Shadow Vault Key, the number of variables X, the base b_i for the Token, and the protocol version.

Since every Vault Entry is paired with a corresponding Shadow Vault Entry, the Recovery Token also exists in the friend's Shadow Vault. If the friend requires Recovery, after receiving it, any Token that had been deposited into the friend's Vault will be Recovered, allowing the friend to then assist any User who has requested Recovery assistance. This enables a chain reaction Recovery process in the viral network of friends.

It is important to note that the VES Recovery chain reaction requires each friend to manually, or semi-manually if such an approach is adopted, enter his/her VESkey at each link for the chain reaction to proceed to the next link (next user). This mandatory human involvement at each link puts control with the humans and keeps the computers, or hackers, from taking over control of the chain reaction, and thus control of the encrypted content. It also creates a firewall between each user's content so that if one user's account has been breached, it cannot be used to set off an automatic chain reaction of breach events using the viral network – assuming Distributed Recovery has been properly set up.

A new Shadow Vault Key is generated by one of two events: (1) whenever a User creates a new VESkey, such as in the case of losing a VESkey; (2) whenever the User manually changes his/her Recovery settings, such as the list of friends or the number of friends, X, required to achieve Recovery. In the second case, the old Shadow Vault Key is immediately and permanently deleted. In the case of a lost VESkey, the old Shadow Vault Key is retained until either the User finds the lost original VESkey and enters it to retrieve the lost information, or Recovery through friends' assistance has been achieved.

When the User loses his/her VESkey, the current Vault Key record is labeled as lost, and a new Vault Key, based on a new manually entered VESkey, is created. All Recovery friends are automatically notified that the User has lost his/her VESkey and needs Recovery assistance.

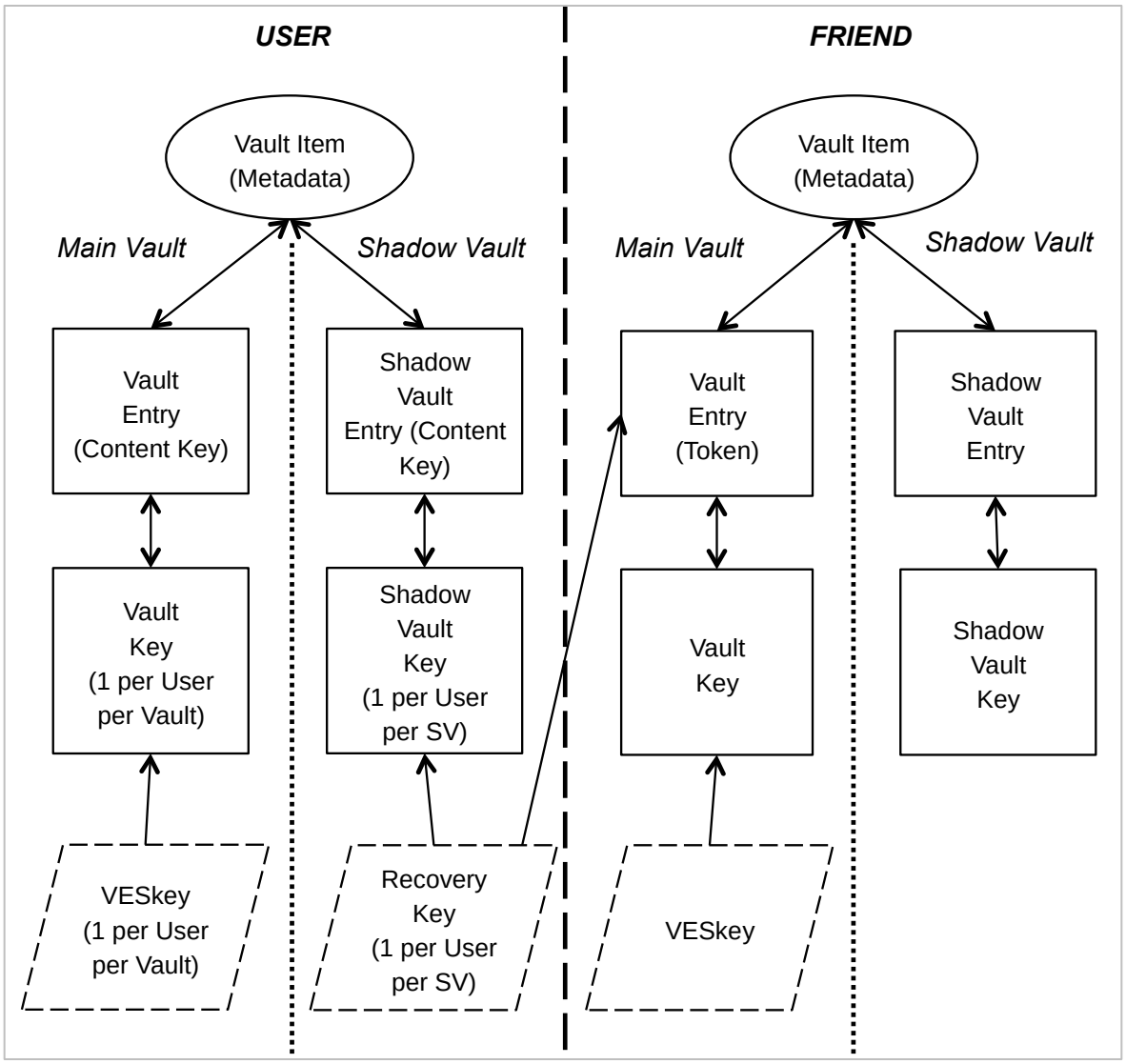
When the friend chooses to assist in Recovery, the Recovery Token in the friend's vault is decrypted through a series of events starting with the friend manually entering his/her VESkey. The open text Recovery Token is then re-encrypted using the public component of the User's recently created Vault Key, and deposited as a new Vault Entry for the User. The User is automatically notified when assistance has occurred.

Once the User has at least X Tokens from the friends who provided assistance, Recovery can be achieved. When the User next enters his/her new VESkey, the system will go through the following steps:

- Decrypt the Recovery Tokens using the unlocked Vault Key for each deposited Token (Vault Entry);
- Use b_i and X from each Vault Item metadata to construct the matrix of coefficients;
- Use the coefficients and the values T_i of each Recovery Token to solve the system of linear equations through Gauss-Jordan reduction, and get the vector of variables $V_1 \dots V_X$;
- Apply chain decryption to $V_1 \dots V_X$ to produce vector $U_1 \dots U_X$, where the Recovery Key $R = U_1$
- Use R to decrypt the private key of the Shadow Vault Key;
- Use the private key to decrypt each Vault Entry pertaining to the Shadow Vault Key;
- Re-encrypt each Vault Entry using the public key of the User's recently created current Vault Key and correspondingly re-deposit the results as new Vault Entries;
- Generate a new Shadow Vault Key, and use it to create the matching new Shadow Vault Entries;

- Distribute the new Recovery Tokens to the friends' vaults;
- Permanently delete the old Shadow Vault Key, as well as all Vault Entries pertaining to it.

At this point, the Recovery process is complete.



This diagram illustrates the database structure and relationships pertaining to a User and one of the friends selected by the User for Recovery. Oval shapes represent open text entries, rectangular shapes represent cipher text entries, parallelogram shapes represent information that is not stored in the database – VESkeys are manually entered by users and Recovery Keys are temporarily created when creating a Shadow Vault Key and reconstructed from Tokens during the Recovery process. This illustration represents relationships prior to the loss of a User's VESkey, when the User has only a single Shadow Vault. The Token stored in the friend's vault, when combined with other Tokens from other friends, (which are not shown on this diagram) will generate the Recovery Key linked to the User's current Shadow Vault.

Vulnerability Assessment

Precursor: Web Based vs. e2e

The traditional understanding between true e2e encryption and any other form of encryption pertains to if the service provider ever has any access to the open text version of the data or access to keys to decrypt the data. Typically, if a service provider executes the encryption algorithms on the server, they not only have access to the open text information, they usually retain access to the keys so that they can later decrypt it for various purposes, including providing a backup if users lose their keys.

The VESvault encryption engine is currently located on the server side and as such, encryption is not true e2e. With the launch of the APIs, users will be able to use the encryption engine on the client side device for true e2e encryption. In the meantime, neither copies of the open text data nor copies of the open text keys are copied or stored on the server. Therefore, while encrypted, VESvault has no access to user content. Only for a brief instant, during encryption and decryption, is the content exposed as open text while on the server. And, the open text keys are immediately deleted once the encryption process is complete.

With the inclusion of e2e encryption, users will be able to use both server side and web-based e2e encryption, interchangeably if they desire, or, they can use e2e exclusively. Because of this, it is worth noting the differences in vulnerability between the two options.

With the web-based solution that employs server side encryption, it is important to consider Transit Vulnerability. This includes the interception of VESkeys or open text data while being processed on the server, or while in transit over data links. The latter is mitigated by using state of the art TLS transport encryption. As for the former, there may always be a possibility of mishandling sensitive data on the servers, whether done intentionally or not. The use of e2e encryption eliminates this problem, rendering the use of e2e VES no more susceptible to Transit Vulnerability than any other e2e service.

Specific services, such as VES encrypted email or text messaging, may benefit in terms of convenience by using the web-based solution instead of e2e. In this case, a separate secondary Vault Key and a secondary VES passphrase may be used, which is specific to the service. In case of a data breach, the extent of the damage will be limited to the data encrypted using the secondary Vault Key, while the main VESvault will still be unaffected. The VESkey will provide Recovery for the main VESvault as well as for the secondary passphrase.

Defining Three Categories of Vulnerability

All decryption keys stored on the server are stored in cipher text format, and the only keys that can decrypt them, or initiate the process through which keys will decrypt other keys, are stored outside the server, outside the Internet and normally outside the client's device. To initiate any decryption process, a human must manually enter a VESkey, or manually enter a portion of the VESkey in the case when the VESkey is derived from two components whereby the long component is stored in the User's devices, and the short component is manually entered by the User. This condition forms the framework for understanding VES vulnerabilities.

For the purposes of comparing the vulnerabilities of VES to currently used end-to-end (e2e) encryption services, it's best to segment vulnerability into three types: Identity Theft, Token Access and Brute Force. Only through these categories of attacks can hackers initially break into the system to gain access to the private encryption keys needed to unlock encrypted content. It is important to note that multiple types of attacks may be necessary in each of these categories for the Hacker to gain access to encrypted content (e.g., Brute Force requires both hacking into the server to gain access to encrypted content as well as brute force decrypting the cipher text).

Identity Theft is defined as a nefarious actor gaining access to either the User's email account, VESvault username and password, 3rd party social media account that was used to create the User's VESvault account, or by gaining access to the User's unlocked device.

There are two types of vulnerabilities pertaining to Token Access that can best be categorized as Authorized or Unauthorized. When selected as friends, the User has given Authorized access to Tokens in that an individual Token will be routed to each friend's device to remove the encryption layer from the friends' VESkey before it is re-encrypted with the public key belonging to the User's account. The friends can use this Authorization to collude with other friends to gain access to the Recovery Key – this is considered Authorized Token Access vulnerability. All other methods of access to Tokens are considered Unauthorized in that any party coming into possession of the Token was not authorized in any way to do so. Both Brute Force and Identity Theft are two ways to gain unauthorized access to Recovery Tokens and are considered under those vulnerability categories, respectively.

To more specifically summarize the three types of attacks: Brute Force hacking of any kind including Tokens, keys or content; Identity Theft which is the stealing of the User's or friends' accounts; and Token Access through the collusion of individuals who have been Authorized as friends by the User.

Brute Force Vulnerability

With VES data being e2e encrypted, there is no way for the service provider, or anyone else who has access to the server or encrypted data, to decrypt the information. In this regard, VES is no more vulnerable than any other e2e encryption storage service.

In general, a successful Brute Force attack on current state-of-the-art encryption is considered to be all but impossible. For such an attack to work on VESvault, it would also need to be successfully combined with another form of attack. In the case of gaining access to the encryption keys, the Hacker would need to hack the system to gain access to the cipher text keys, then Brute Force decrypt at least one key, then possibly hack into the system containing the encrypted content to use the key on the content. That's one highly improbable Brute Force attack in conjunction with at least one very improbable server hack, which makes this attack very unlikely, adding no more vulnerability than stored e2e data without VES.

The Hacker may also attempt to brute force hack the Tokens. This would involve the same string of attacks as the above described brute force attack on the keys, but with additional brute force decryptions. The Hacker would need multiple Tokens to reconstruct the Recovery Key and would need to successfully execute as many independent brute force attacks as the number of Tokens needed for Recovery. Thus, this attack is much less likely than the brute force attack on the keys themselves, with negligible associated risk.

Identity Theft Vulnerability

There are four types of Identity Theft Vulnerability: Device Theft, VESvault Account Theft, Email Account Theft and 3rd Party Account Theft.

Device Theft involves a nefarious actor gaining physical access to the User's electronic device through which the User accesses his/her VESvault account – this type of criminal is more a burglar or robber than a hacker. Email Account Theft involves a hacker actor gaining remote access to the User's email account that was used as the username for the User's VESvault account. 3rd Party Account Theft involves gaining remote access to the User's 3rd Party social media account (such as Google, LinkedIn or Facebook) that was used to create the User's VESvault account.

Starting with Device Theft, since the stolen device can be a cell phone, which acts as two devices in terms of 2 factor authentication, the theft of the phone has a higher level of risk than theft of a computer or tablet that is not used as the 2nd factor of authentication.

The base assumption in the stolen phone, tablet or computer is that the PIN or password must also be stolen, or the device must be unlocked when stolen, giving the Hacker access to the User's apps and content.

In the case of a tablet or computer stolen without knowledge of any passwords or PINs, the worse case scenario is that the device was left in an unlocked state and all passwords have been stored in the browser, allowing the Hacker to access any account to any app with a stored password. This includes VES. However, even in

this situation, there is a very good chance the Hacker will not have access to the User's encrypted content in VESvault.

A two-step process is required to access all encrypted content in VESvault. The User must first log into VESvault with his/her password – which may be stored in the browser – but this doesn't open the access to the encrypted content. The User must enter his/her VESkey to unlock the encrypted content. The VESkey is different from the password. By default, the VESkey is not stored in the browser and the User must manually enter it for each browser session, and often multiple times during a single browser session. Since there is also a short inactivity timer that automatically closes the encrypted vault, in the amount of time it takes for the User to walk away from his/her device to the time the thief would be in a position to comfortably access the User's VESvault account, it is unlikely that the encrypted section will be open even if there is an open VESvault browser session.

Whether the VESvault browser session is open or not, the Thief needs the VESkey to open the encrypted section. Without it, the only alternative is to initiate Recovery, which requires the password as well as 2nd factor authentication. Without the password and cell phone, the Thief cannot initiate Recovery and without Recovery, he cannot get a new VESkey with which to access the encrypted section.

If the Thief is unable to retrieve the password from the device, the Thief could attempt a password reset, but this also requires 2nd factor authentication, which means it would only work if the stolen device were the User's cell phone.

In summary, the Thief must steal the User's cell phone with the phone unlocked, to open a VESvault session, then initiate a password reset using the phone to bypass the 2nd factor authentication, then initiate Recovery, again using the phone to bypass the 2nd factor authentication. At this point, the Thief still will not have access to the User's encrypted content because VESvault employs a unique 3rd factor authentication. (It is worth noting that by obtaining the unlocked cell phone, the Thief would already have access to all encrypted information for any other e2e

app that stores the encryption key, but not VES encrypted content. In this regard, VES is more secure than these other e2e services.)

The unique, VES 3rd factor authentication is that friends are supposed to, and can be required to, voice or video verify the person seeking Recovery is the actual User. This is not stored information. Rather, it is real time decision-making and real time external human control that exists outside the Internet. It cannot be hacked without obtaining the VESkeys of *X* of the friends needed to achieve Recovery, all of which exist outside the Internet. If a friend doesn't believe the request is genuine, the friend does not manually enter his/her VESkey and the Hacker will not receive the decrypted Token. This combination of external human decision-making and control makes VES a unique, independent, 3rd factor of authentication, different from the commonly cited factors: *something you know* (password), *something you have* (cell phone) and *something you are* (fingerprint).

It's worth noting that what's traditionally been labeled as a 3rd factor of authentication, *something you are,* is not really independent from the first factor, *something you know,* in that both are discrete pieces of semi-static information supposed to be in the exclusive possession of the User, but both are passive information that can be intercepted and stored for a later, remote hack. In this regard, a fingerprint becomes no more useful than a compromised password that you can never change, becoming forever useless. In this consideration, the VES 3rd factor authentication becomes more useful in that it is not information that can be conceivably intercepted and stored, nor is it static or unchangeable. As an aside, the applicability of VES 3rd factor authentication could extend beyond VES Recovery to be used as a means for account recovery for any app or SAAS, should a user forget their password, PIN or otherwise access to an account, regardless if the user's content is encrypted or not.

Additionally, the greater the number of required friends the User has set up to achieve Recovery, the greater number of friends the Thief will need to bypass or fool in VES 3rd factor authentication. Also, friends will be able to set off an alert during the verification process if they think the request is a hack. The alert

immediately goes to all other friends as well as the User's email account and phone number. In the case of requiring verification prior to assistance, Recovery can be set up through a communication app that requires a successful video or voice call.

VESvault Account Theft is the remote stealing of the User's VESvault username and password without access to any of the User's personal devices. This is different from stealing the User's 3rd party social media account that had been used to setup the User's VESvault account.

Even if the Hacker knows the VESvault password, he could not access the encrypted section without the VESkey. Without the cell phone, the Hacker would not be able to bypass the 2nd factor authentication process needed to initiate Recovery. And, even if he could, he would also need to circumvent VES 3rd factor authentication.

Email Account Theft occurs when the Hacker remotely gains access to the User's email account and can access the User's inbox, without having physical access to the User's device. Assuming the email address is the same as the username for VESvault, the Hacker would still not be in possession of the VESvault password and could not execute a password reset without the cell phone for the 2nd factor of authentication. Even if the Hacker had the password, he would still need possession of the cell phone, or other source for the 2nd factor of authentication, to initiate Recovery because the Hacker does not have possession of the VESkey. This doesn't even consider getting by the VES 3rd factor authentication. A remote email account hack would be ineffective.

Remote 3rd party social media theft is even less effective as remote email account theft because, in addition to all the other obstacles in the remote email attack, the hacker would not be able to receive the email link for the password reset. A remote social media account hack would be ineffective.

Token Access Vulnerability

This last vulnerability is unique to VES and not possible with other existing e2e services.

Other than a hack into the server along with brute force decryption, there are two ways to gain access to Tokens. The first way is to be selected as a Recovery friend by the User. The second is to install malware on the friend's device that can read all keystrokes and capture content when in an unencrypted state.

Addressing the malware approach, if malware were available to capture the Tokens as they are decrypted during Recovery, the malware would have already captured the friend's VESkey and would have already had access to the friend's encrypted content before ever having the means to gain access to the User's encrypted content. For this reason, the malware attack is more appropriately considered Identity Theft. As such, VES is no more vulnerable than any e2e encryption without VES.

The other method of Token Access is for a number of colluding con artists to con the User into selecting them as friends. The con artists would also need access to expert hacking skills to hack into the server to gain access to the Tokens, the Shadow Vault Key and all the keys stored as Vault Entries – in essence, the Hacker would need to extract the entire encrypted database or filter through multiple areas to find all information pertaining to the User. With the Tokens in hand, the nefarious actors would then need to descramble the Tokens to recreate the Recovery Key, decrypt the keys, and then, in the usual case when the encrypted content is stored with other service providers and hence other servers, hack into this additional server to steal the encrypted content and decrypt it. Alternatively, the colluding actors would need to gain access to the User's VESvault account to complete the Recovery process, and thus commit Identify Theft in addition to the collusion.

In the case of e2e encryption for VES, friends do not gain possession of Tokens until either the User or the friend initiates Recovery. When the User initiates

Recovery, the system allows the friend to access the deposited Token in that friend's vault for decryption. When the Friend initiates and receives Recovery from his/her friends, the contents of the Friend's Shadow Vault are accessed by the Friend's device to be decrypted with the Recovery Key. Included in this package is the backup copy of the User's Token. In this case, a Hacker could control when the User's Token is sent to his client side device by initiating his own Recovery. However, this situation can be greatly mitigated if a new Shadow Vault Key for the User were automatically generated the next time the User enters his/her VESkey after any friend has gone through a Recovery process.

Regardless of how the Hacker gets possession of the Token, the success of this chain of events is extremely unlikely as the list of possible hackers would be limited to the number of friends the User selects for Recovery, involve an elaborate personal con with multiple colluding actors, and require at least one but more likely two successful hacks into two independent separate servers with state of the art cyber security. And, this entire effort will only gain the encrypted content of one single person and not a massive data breach for multiple users.