VESwallet

The integration of VES APIs with the open source code from MyEtherWallet/MyCrypto

Overview

VES API integration with MyEtherWallet

VESwalletuses VES to provide safe and reliable recovery of the wallet password, separate from the storage and backup of the encrypted private key that it decrypts. This improves and expands the Keystore storage option with three increasingly secure use cases that have advantages compared to: 1) the traditional Keystore option; 2) the traditional MetaMask and MyEtherWallet (MEW/MyCrypto) Chrome extension options: and 3) traditional cold wallet options.

The benefits of VES to wallet holders

In the VES equivalent of all three use-cases, the password that decrypts the private key is safely encrypted and backed up to the cloud using VES, reducing the risk of losing it and allowing safer, separate storage of multiple local copies of the encrypted private key. The result is lower risk of wallet loss through key loss and the elimination of a possible single point of vulnerability that exists with the legacy versions of the Keystore, MetaMask and cold wallet key storage options in co-location of the password and encryption key.

VES: Viral Encrypted Security

VES enables encryption to be safer without degrading privacy or security. It employs a viral network of friends to enable the recovery of encrypted content in the event the owner loses the passphrase. The base engine in VES is similar to Shamir's Secret Sharing; VES uses linear algebra while Shamir's uses polynomials. More importantly, VES has overcome the two limitations of past implementations of Shamir's that have kept Shamir's from being practical: reliability and vulnerability. With just a small network, VES can easily achieve a

recovery reliability of well over 99.99999%**. And, VES mitigates the friend collusion problem inherent with Shamir's. Check out the *How VES Works* section for more information on VES functionality. For more information on VES reliability, check out our FUNMATH interactive probability calculator www.vesvault.com/fun-math.

A focused, limited implementation of VES

VESvault Corp. is not in the digital wallet business. Through our licensing agreement, we will offer our APIs to any SaaS that wants to extend the benefits of VES to their customers. To test our beta APIs, we decided to do the first integration ourselves and chose MEW/MyCrypto. We chose to limit VES integration to the Keystore option. We have submitted our code revisions through Github to both MyEtherWallet and MyCrypto and hope they take over from here.

Impact of the broader applicability of VES

VES works with any stored encrypted content: digital wallets, documents, pictures, videos, emails, group texts, vmails, audio files, medical records, etc. VES will also work with the future decentralized blockchain storage ecosystems, as well as traditional cloud-based and local storage solutions. Complementing this breadth of applicability, a user's VES network is universally applicable across all their apps. If a user creates a VES network with VESwallet, that same VES network works with any other VES enabled app. And, with the addition of each new friend to the user's VES network, that friend's entire VES network is automatically appended to the user's network, further strengthening VES recovery for the user.

---

* VESvault Corp. does not guarantee any level of reliability of Recovery nor that Recovery will occur at all. The probability of Recovery is 100% determined by the depth, breadth and quality of the Friends selected in each User's VES network. VESvault Corp. does not and cannot affect the quality of Recovery.

Three Keystore Use-Cases

Universal across all three use-cases

In all three VES-based Keystore use-cases, VES is used to store/backup the password that decrypts locally stored versions of the private key. This eliminates the need to store a local paper, electronic or hardware copy of the password. This is generally safer because there's less chance of losing the password. It's generally more secure too in that if a hard copy of the password is compromised, it usually occurs at the same location as the computer storing the private key, enabling possible access to the wallet. It also means that copies of the encrypted private key no longer have any security risk due to co-location with the password, and many more copies can be safely stored in less secure locations (multiple drawers, filing cabinet, shoe box, car, work, etc.). Overall, using VES to backup the password can make the wallet *safer* from key loss and more *secure* from an attack.

The wallet password is encrypted with a chain of Vault Keys that can be unlocked with the user's VESkey, which is also a passphrase. If the user forgets the wallet password, or choses to not memorize it at all, they can manually enter their VESkey inside a VESwallet popup to retrieve it and automatically populate the wallet password entry field. Since the VESkey itself is backed up by VES, if the user loses their VESkey they can initiate VES recovery from VESvault.com to restore their VES level access, which will subsequently restore access to all the VES apps including the wallet password. Thus, with a properly configured VES network, the user has a high degree of assurance their VES encrypted wallet key will never be lost, hacked, damaged or stolen.

In all three use-cases, comparisons are made in terms of *security*, the risk of the wallet being compromised to a nefarious actor, and *safety*, the risk of losing the wallet.

User case 1: stand alone Keystore option

The standard Keystore option of using MEW/MyCrypto through an online browser is not recommended by the makers of MEW/MyCrypto because of phishing risk. To neutralize this risk, they recommend bookmarking the official MEW/MyCrypto website to ensure the user never visits a malicious website masquerading as MEW/MyCrypto. VESwallet does not diminish this phishing risk. However, if the user bookmarks the official site, this risk is mitigated.

One advantage that the VESwallet Keystore option has over the legacy version pertains to eliminating the single point of vulnerability, making it more *secure* than the legacy option. Storing the password in VES while storing the encrypted private key locally eliminates this security risk. It also allows the user to make many backup copies of the encrypted private key because there is no fear that a nefarious actor can steal the wallet by gaining access to a USB drive containing a single point vulnerability. This higher level of redundancy for the locally stored private key, along with the safety of VES for the password, significantly reduces the risk of losing the wallet and creating an advantage in *safety* over the legacy Keystore option.

User case 2: MetaMask and MEW/MyCrypto Chrome extension

To neutralize the possibility of visiting a phishing site, the makers of MEW/MyCrypto recommend MetaMask as a private key storage option. Since it is a browser extension, the user interacts locally with the private key and never enters the password into a website. Like MetaMask, MEW/MyCrypto is also offered as a Chrome extension to accomplish the same thing. While MetaMask stores the encrypted

private key locally, within the browser, the MEW/MyCrypto Chrome extension stores the encrypted private key in the cloud.

MetaMask has a Seed that can be used to recreate the wallet. The Seed will recreate every wallet, and hence every private key, that was created in the same MetaMask browser extension. This recovery provides for the contingency against losing either the hardware with the MetaMask Chrome extension or the password used to unlock the private key.

MetaMasks has drawbacks in both safety and security. The Seed is a single point vulnerability for MetaMask, a security issue. Also, the private key is only stored in one place (the browser). The user will lose the wallet completely if either the password and Seed is lost, or if the computer and Seed is lost. Since there is only one computer and very few copies of the Seed, this results in very low redundancy of either the private key or Seed – a large safety vulnerability. Lastly, the Seed recovers all the private keys created in MetaMask so multiple wallets are at risk if the Seed is compromised – another security issue.

The disadvantages of the MEW/MyCrypto Chrome extension are the same as those of MetaMask, except that having the encrypted private key stored in the cloud reduces the risk of losing it (safety). However, this comes with a cost in security in that the encrypted private key is online and possibly encrypted with a weak password, as passwords are manually created by users and not randomly generated.

A future Chrome extension version of VESwallet would match the security benefits of MetaMask and MEW/MyCrypto Chrome extension for phishing attacks. While doing so, it would it would eliminate the single point safety and security vulnerabilities outlined above. Likewise, the current web version of VESwallet equals these safety and security

benefits provided the user only uses a bookmark to access the site and to prevent phishing attacks.

Another option is to create the wallet using VESwallet and backup the password to VESvault, and then to only use an offline version of MEW/MyCrypto on an offline computer to do all wallet transactions. If the password is ever lost, the user can recover it online using VESvault and then go back to using the offline setup. This option provides additional protection against phishing attacks.

User case 3: cold wallet

The cold wallet solution for legacy MEW/MyCrypto improves upon the MetaMask option by the mitigation of malware attacks. Encryption and decryption occur within the closed operating system of the cold wallet, and malware that reads keystrokes, screens or any other computer activity is ineffective at accessing the private key.

There is a legacy home brewed version of MEW/MyCrypto that can mimic a cold wallet and mitigate the malware problem. The legacy home brewed approach is to install an isolated operating system on a reclaimed computer to be used specifically for wallet transactions. The suggested operating system is *Tails*. An offline version of MEW/MyCrypto is then installed on the Tails computer. All wallet transactions are conducted on this isolated computer, mimicking the same level of security as a cold wallet. The signed transaction is transferred to a storage device, which is then connected to an online computer to upload to the blockchain.

A similar approach can be used with VES. After creating the wallet with VESwallet and saving the wallet password to VESvault, the process steps are identical to the legacy home brewed version. The offline computer running the Tails operating system and offline version of

MEW/MyCrypto is used to create the transaction, which is then saved to a storage device. The storage device is then connected to an online computer to upload to the blockchain. If the password is ever lost, the user can use VESvault to recover it.

A traditional cold wallet is not without its flaws. First, there are generally no duplicates of the hardware itself. The private key is stored in only one place and that's on the locally kept wallet hardware. If the hardware is lost, the private key is lost and the only means of recovery is to use the Seed. The wallet is protected by a PIN that if lost, the private key becomes lost. In summary, if either the PIN or hardware is lost, the private key is lost and the only means of recovery is the Seed.

The purpose of the Seed is to completely recover the wallet in the event that either the hardware wallet or PIN is lost. The Seed, usually written on paper and not stored electronically unless on a USB device, is used online so that anyone, with nothing else, can recover the wallet from any location through internet access. Thus, the Seed is a single point of vulnerability and must be kept extremely secure. At the same time, the Seed is the only backup and must be reproducible with a high degree of confidence. These are conflicting needs. The user would want few copies of the Seed to keep it secure and multiple copies to ensure it is never lost. This requires the use of multiple safes, or safe deposit boxes, which is not commonly available to many users.

The home brewed cold wallet version using VES improves upon the traditional cold wallet in a number of ways. First, it eliminates the single point of vulnerability in security. In the unlikely event that the wallet password is hacked, without access to the locally encrypted private key, the wallet is still safe. Conversely, if someone were to gain access to the device with the encrypted private key, it would be useless without the wallet password.

Second, there is more redundancy in backup, making it safer from loss. Where there is a single hardware device for a cold wallet, there can be many USB drives, computer drives, phones and other drives that store the encrypted private key. With a proper VES network backing up the wallet password, the weakest link in recovery becomes the number of local devices that store the encrypted private key. Since the encrypted private key is not a single point of vulnerability in security, and isn't overly prohibitive to duplicate, it can be stored in much greater numbers than either a single hardware wallet or two or three paper copies of a Seed.

How VES Works

Illustration for VES recovery & wallet password retrieval

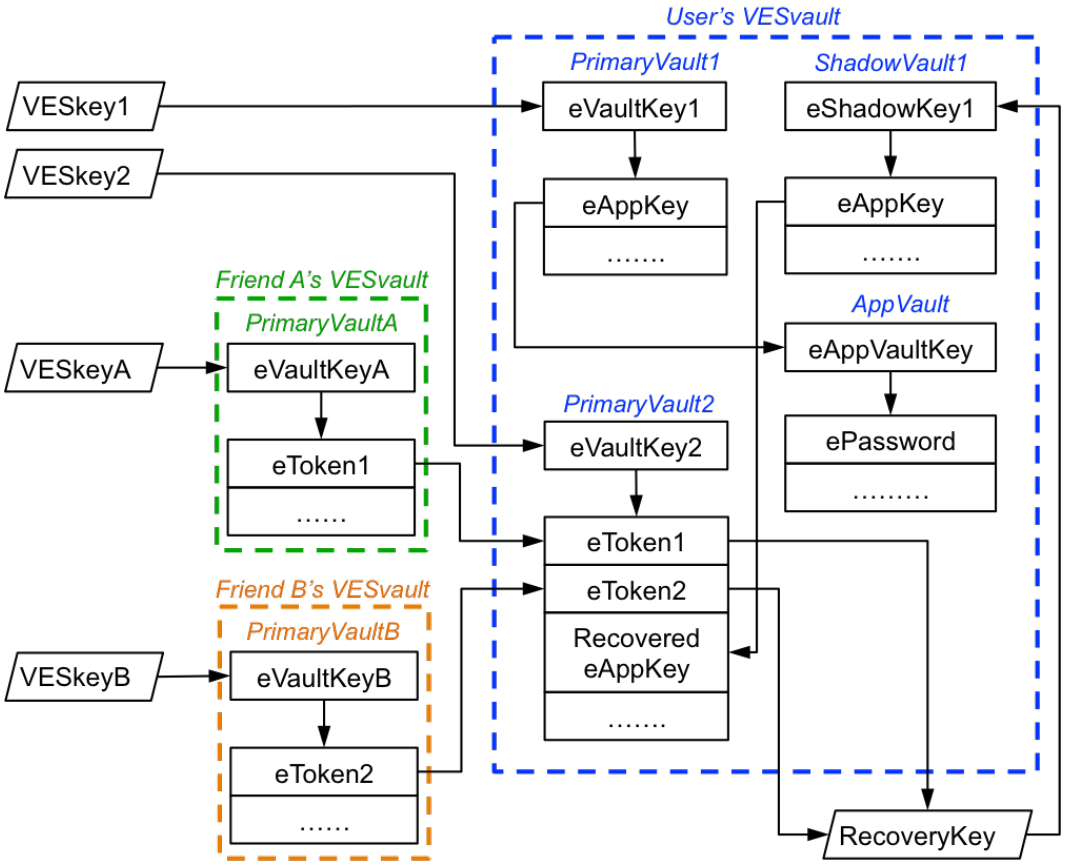

Illustration overview

The above figure is meant to show how VES recovery and wallet password retrieval work (and hence how any app works with VESvault). As such, some system components, process steps and relationships that are not critical to the explanation may be missing or simplified. All encryption and decryption steps occur on the client side, making for full end-to-end encryption.

Illustration Conventions:

- An "*e*" in front of any label indicates that the item has been encrypted. For example, *eVaultKey1* is the encrypted version of *VaultKey1*
- All keys in the parallelogram shapes are symmetric encryption keys – all the *VESkeys* and the *RecoveryKey*. These keys are never stored on a computer: *VESkeys* are manually entered by users, and *RecoveryKey* is temporarily recreated by the descrambling process using the *Tokens.*
- All keys in rectangular shapes are asymmetric encryption keys – all *VaultKey* entries and the *ShadowKey1*.
- All arrows originate from keys and show the items that they decrypt, either directly or indirectly, as is the case with *RecoveryKey*.
- Every *PrimaryVault* normally has a *ShadowVault*, but only *ShadowVault1* is shown because it is the only *ShadowVault* used in the recovery process.
- Everything within an individual user's Vault is accessible only through either the user's VESkey(s) or RecoveryKey(s).

Depositing and retrieving a wallet password from VESvault

Before a User can save their wallet password to VESvault, they must first create a VES account. This step is built into the process of saving the wallet password with VES in VESwallet, if the VESvault account is not already created and logged into. Once the VESvault account is setup and open, the User can then finish the process by confirming that the wallet app has access to the VESvault account.

Once the VESvault setup process is complete, a random symmetric AppKey and asymmetric AppVaultKey are created, with the private

component of AppVaultKey encrypted by AppKey to form eAppVaultKey. AppKey is encrypted by the public component of VaultKey1 and Shadow Key1 to form the eAppKeys deposited in PrimaryVault1 and ShadowVault1, respectively. The public component of AppVaultKey is used by other users to make deposits to the User's AppVault (i.e., this would work for non-wallet apps, such as emails). The AppVault is where all encrypted records for the app are stored, including the wallet password, which is stored as ePassword.

If the User either forgets the wallet password or simply chooses to use VES instead of manually typing it in, the User selects the *Retrieve your wallet password with VES* button in VESwallet and then manually types in *VESkey1. VESkey1* is then used to decrypt *eVaultKey1* to create *VaultKey1*, which is then used to decrypt *eAppKey, which in turn decrypts eAppVaultKey, which in turn decrypts ePassword to generate Password* and populate the password entry box in VESwallet.

VES recovery

For the wallet password retrieval process, the VESkey is used to retrieve the wallet password (or any app password), while VES recovery is when the User loses their VESkey and is in need of recovery.

VES recovery must be set up prior to the loss of the VESkey. The User selects a small number of friends, *N*, to call upon to assist in recovery. In the illustration, *N* is not known nor is it shown. The User also selects the number of friends required, *X*, to assist the User to achieve recovery. In the illustration $X = 2$.

For every *PrimaryVault* there is a matching *ShadowVault* with the same items, but the only *ShadowVault* shown in the illustration is *ShadowVault1.* Each *ShadowVault* has a separate key from its

matching *PrimaryVault*. In the case of *ShadowVault1*, the encryption key is *ShadowKey1* while matching *PrimaryVault1* has encryption key *VaultKey1*.

While the Password retrieval process goes through *PrimaryVault1*, the VESkey recovery process uses *ShadowVault1*.

With *N* and *X* established, VES creates *N* scrambled independent linear equations, each with *X* variables, and allocates a unique token pertaining to each equation to each of the *N* friends. In this case there are two tokens, *Token1* and *Token2*. *Token1* is encrypted with the public component of *VaultKeyA* to create *eToken1*, and *Token2* is encrypted with the public component of *VaultKeyB* to create *eToken2.* Both *eToken1*and *eToken2* are stored in the Primary Vaults and Shadow Vaults (not shown) of *Friend A* and *Friend B,* respectively.

The link, *I Lost My VESkey*, is available at every VESkey entry box and the User selects it when *VESkey1* becomes lost. Upon doing so, the User is required to manually create *VESkey2.* In doing so *PrimaryVault2* is also automatically created. Simultaneously, an alert is sent out to the User's friends.

Friend A and *Friend B* can assist the User by entering *VESkeyA* and *VESkeyB*, respectively, along with selecting the *Assist* button in VESvault. *VESkeyA* and *VESkeyB* then decrypt *eVaultKeyA* and *eVaultyKeyB*, respectively, which in turn decrypt the *eToken1* and *eToken2*, respectively. (Separately, *Token1* and *Token2* are unusable as keys and are essentially as difficult to unscramble as it is to brute force hack state of the art encryption.) *Token1* and *Token2* are then re-encrypted using the public component of User's new asymmetric key, *VaultKey2*, and deposited in the User's new *PrimaryVault2*. The next time the User enters *VESkey2* to access the new *PrimaryVault2,*

VESkey2 decrypts *eToken1* and *eToken2* from *PrimaryVault2. Token1* and *Token2* are then descrambled to re-create *RecoveryKey*, which is used to decrypt *eShadowKey1* to re-create *ShadowKey1*. *ShadowKey1* is used to decrypt all the contents of *ShadowVault1,* including the backup copy of *eAppKey*. All the decrypted contents of *ShadowVault1* are then re-encrypted with the public component of *VaultKey2* and deposited into *PrimaryVault2*. This completes the recovery process and the User can now use the new *VESkey2* to access the wallet *Password*. At this point *PrimaryVault1* and *ShadowVault1* are deleted.

Usability considerations

When looking at this process, it is important to note why *Password* and any VESkey, such as *VESkey1* or *VESkey2*, should not be the same. Unlike AppKeys, user created VESkeys are never shared with any app. The recovery process is unique to VESvault. This provides a safeguard in that while the app can do anything it wants with the AppKey and the contents of the App Vault (any 3rd party app developer can do anything they want with their own information), it can do nothing with the information in the VESvault. This allows VESvault to introduce additional security measures.