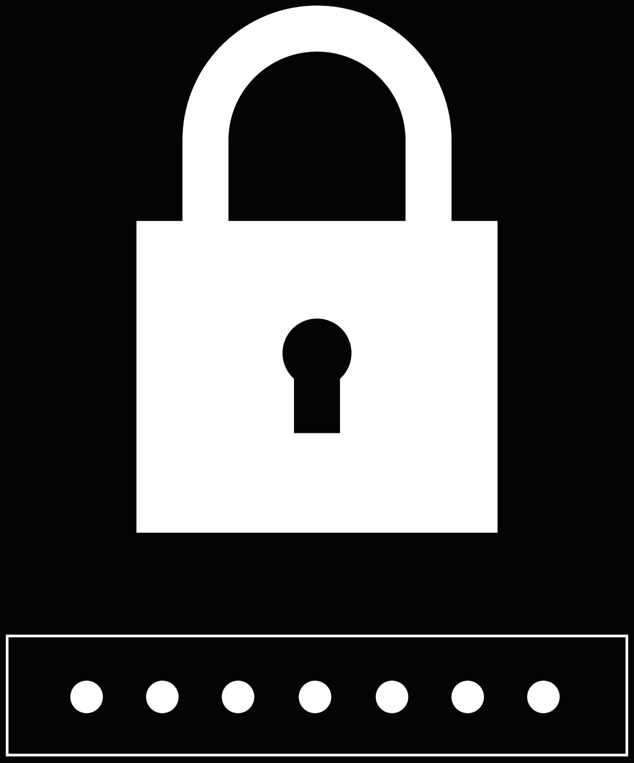
RPass Security Whitepaper

RPass Security Design: Enterprise-Grade Password Management for Teams



RIPPLING

Table of Contents

Table of Contents 2

How RPass makes your company more secure 3

RPass security principles 4

Rippling security overview 6

Individual Password Management 6

Team Password Management 6

Encryption fundamentals 7

RPass security design 10

About the Master Password and Master Key 10

Personal Vault. 11

Company Vault 11

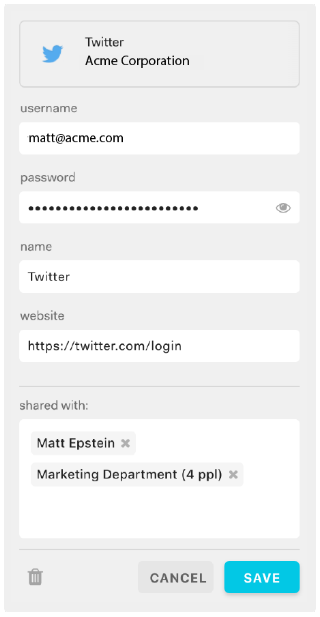
Upgrading Encryption Options 12

Server Communication 12

Two-Factor Authentication. 12

Use of Web Crypto 12

Terminology (Appendix). 13


Twitter
Acme Corporation

username: matt@acme.com
password: ••••••••••••••••••••••••
name: Twitter
website: https://twitter.com/login
shared with: Matt Epstein; Marketing Department (4 ppl)

RPass in action: *What your employees will see when they share their password with fellow coworkers.*

How RPass makes your company more secure

Passwords are the cornerstone of your IT security, and RPass ensures that your data is secured by our enterprise-grade cryptography and technology standards.

Store Passwords in a Zero-Knowledge Vault
RPass encrypts passwords before they even leave the user's device, and Rippling itself never has access to the decryption keys. So Rippling has zero knowledge of the unencrypted passwords.

Share Passwords Securely
RPass eliminates password sharing on Post-Its, email, chat sessions and other insecure methods. Users safely share passwords directly from the RPass software, which transfers them from client to client using secure encrypted communication.

Automatically Grant and Revoke Company Vault Access by Role
One of the most common ways for a password to be leaked is when an employee leaves a company or changes roles but someone forgets to shut off their password manager access. Because RPass is tightly integrated with Rippling's HR and onboarding/offboarding system, employees that leave or change roles will automatically be unable to open the company vault.

Encourage Strong Passwords
Users often prefer short, simple passwords so that they can easily remember them. Since users no longer need to remember passwords, they can use longer more complex ones and store them in a protected RPass vault. RPass is available directly in their browser, so passwords travel with users wherever they go.

Protect Against Phishing
The most common data breach occurs when employees enter passwords on phishing sites that look and act similarly to official sites. RPass only fills passwords into the same website domain from which they were saved, so employees can't inadvertently enter passwords on a phishing site. Similarly, RPass won't fill passwords on insecure HTTP sites.

RPass security principles

RPass is a zero-knowledge password manager, meaning its technology is based on the highest level of data encryption while still protecting your privacy by having no access to your sensitive information. In an age where privacy is a major concern for consumers, providing an unparalleled data protection application while still maintaining privacy is one of Rippling's biggest priorities.

True End-to-End Encryption
All communication between client devices and Rippling servers is encrypted with the latest 256-bit cryptography standards.

Private Password Ignorance
Rippling never has access to your data. You maintain your private master keys and passwords, and we store them in encrypted form without having access to decrypted versions.

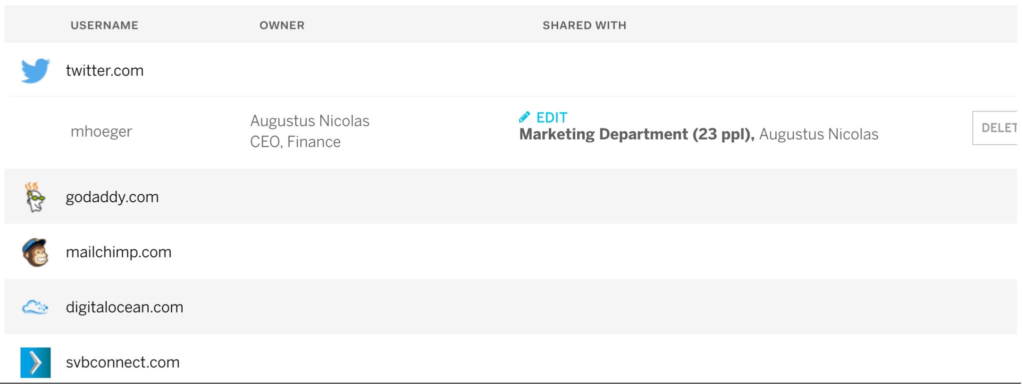
Reliability
Your passwords are always available regardless of your location or device. All you need is access to the Internet.

Brute-force Attack Protection
Even if an attacker is able to steal your encrypted data on a local storage device, our service detects brute-force password attacks and rate limits log-in attempts, which stops an attacker from guessing your password.

Centralized Organization for Better Resource Management
Rippling provides a centralized administration dashboard for easy management of corporate accounts. Whether you add or delete an account, it's a no-hassle one-step process that keeps your password storage access organized.

Security Best Practices
Rippling keeps its security standards up-to-date with the latest industry best practices and cryptography technology that protects from common password attacks.

RPass in action: *Admins have full visibility and control over passwords employees store in their company vault, but no visibility into passwords stored in their personal vault.*



USERNAME	OWNER	SHARED WITH	
twitter.com			
mhoeger	Augustus Nicolas CEO, Finance	EDIT **Marketing Department (23 ppl),** Augustus Nicolas	DELET
godaddy.com			
mailchimp.com			
digitalocean.com			
svbconnect.com			

RPass Security Overview

A (somewhat) non-technical overview of our security design.

Rippling Security Overview

Security and privacy of your data is our highest priority, and our comprehensive approach reflects this.

Individual Password Management
RPass is a password vault: a private, encrypted storage solution that uses a personalized master key to encrypt and decrypt data. With RPass, encrypted data (saved passwords) is stored safely on Rippling's servers to make them available from the cloud. Using a user's password, a master key is created during initial creation of the vault. This master key is the only value that can unlock stored password.

Team Password Management
RPass also provides enterprise-grade password sharing via password vaults. You can create, share, and remove employee passwords and access to vaults. The company vault is protected with a company vault key. Rippling uses public key cryptography to transfer the company vault key to a new employee in a way that prevents Rippling from learning it.

Public key cryptography has been around for decades, so it's a reliable way to share private data in encrypted form by only allowing the owner of a private key access to a message even if this message is posted publicly. This standard is how RPass customers are able to share passwords within their company while still maintaining privacy. Every user creates their own private key when they join an RPass-enabled company, but the RPass administrator controls access to the company vault.

Using RPass, a business can manage user accounts but still allow employees to have their own private keys and easily share passwords within the corporate environment. Data is still encrypted and protected by Rippling's servers in the same way as an individual account, but only administrators are able to add and remove accounts to maintain the integrity of corporate data.

Security is at the Heart of What Rippling Does

- All data is transferred using 256-bit TLS encryption — the state-of-the-art used by banks and governments.
- Military-grade AES encryption protects your data at rest.
- We follow industry best practices for defense in depth: data is encrypted with multiple keys, keys are rotated regularly, and sensitive data uses end-to-end encryption.

A Strong Team Enables Strong Security

Security and privacy of your data is our highest priority, and our comprehensive approach reflects this.

- We keep our team up-to-date on the latest security practices with regular security and privacy awareness training.
- New features go through extensive testing and peer review with a rigorous SDLC.
- Admin access requires a strong password with two-factor authentication, and separation of duties is built in to sensitive tasks.
- Security teams work around the clock to protect your data and respond to threats.

Tested and Trusted

- We work with independent third-parties as well as external researchers who regularly assess our site for vulnerabilities.

- All data is hosted and processed in an SSAE 16 SOC2 compliant data center, with 24/7 physical security.
- We proactively identify and fix issues with a bug bounty program.

Rippling Makes Your Whole Org More Secure

- Easily enable two-factor authentication for all of your services.
- Use Rippling's hardware management for sophisticated endpoint protection.
- Quickly remove access for ex-employees.
- Download audit logs of Rippling app access.

Encryption Fundamentals

Hashing

To understand the way Rippling safely stores your data, you must understand the basics of hashing.

Hashing produces a deterministic, non-reversible, and mostly unique "fingerprint" of data. Just as it's very unlikely that two different people have the same fingerprints, it's very unlikely that two different input values produce the same hashed value. And given a hashed value, it's very hard to determine the original data that produced the hash.

Using a security hashing algorithm, you hash a value that is later stored in a database. When a user enters the same value, the same hashed value stored in the database is compared with the value entered by the user. If the string is the same, you know that the user has entered a correct value. It's unlikely that two different values return the same hash value, and this is why it's used as a storage solution for sensitive data.

One common use of hashing is to store login credentials for authentication. Rather than storing your password in plain text or using a two-way encryption standard (similar to symmetric encryption described below), a site should store a hash of your password, which makes it a one-way encrypted value.

When you input a password to log in, the site does a hash of your submitted password and compares it to the hash stored in its database. If the hashes match, then by the deterministic and mostly unique properties described above, you must have input the same password that was used to generate the original hash. If a hacker gets access to the database, they can read the hash of your password, but by the non-reversible property described above, the hacker would not be able to calculate the actual password easily given the hash.

For a secure hash algorithm, the only way to calculate the actual password would be to try hashing lots of password guesses (often, billions) until you find one that produces the same hash value. This is called a brute force approach to password cracking.

Symmetric Encryption

Symmetric encryption is a two-way encryption process where the same key is used to encrypt and decrypt values. Unlike asymmetric encryption where a public and private key is used, symmetric encryption uses the same key for both actions. If an attacker has access to encrypted data but not the decryption key, brute-force attacks using billions of guesses are needed to decrypt the data.

Some common symmetric encryption algorithms are 3DES (old), AES128, AES192, AES256, and Blowfish. RPass uses AES256, which is the algorithm certified by the US military for encryption of Top Secret information.

Slower Can Be Better

We're used to thinking that faster is better, but cryptography is one area where some algorithms are intentionally slow. For instance, you don't want hashing to be too fast, otherwise it would be easy for an attacker to reverse your hashes by testing several guesses very quickly.

Many hashing algorithms actually run data through multiple hashes to make it slower. PBKDF2 with SHA256 is an algorithm used extensively in RPass that lets you specify how many rounds of SHA256 hashing to run. More rounds make it slower for the user but also harder for attackers to crack by brute-force.

Public-Private Key Encryption

Public-Private Key encryption (also called asymmetric encryption) is a two-way process where different keys are used for encryption and decryption.

When you generate asymmetric encryption keys you actually generate a pair of keys: a public key used for encryption, and a private key used to decrypt any data that was encrypted using the corresponding public key.

Let's suppose Bob wants to send a message to Alice. Bob wants to store the message on Rippling's server so Alice can retrieve it later, but Bob doesn't want Rippling to be able to read the message. Alice first creates a private key and a public key. Alice shares her public key freely, but keeps her private key to herself. Bob encrypts his message using Alice's public key, so that only someone with Alice's private key can decrypt it. Bob then stores the encrypted message on Rippling's server.

Since Rippling doesn't have Alice's private key, Rippling can't read the message. But when Alice retrieves the encrypted message later, she can then decrypt it with her private key and read the message. This is why asymmetric encryption using public and private keys has become a standard for users who must communicate on a network filled with people who could be eavesdropping.

Regardless of visibility, a message is only accessible using a safely stored private key.

Some common asymmetric encryption algorithms are RSA and ECDH. RPass uses RSA.

Please see "Overview" to understand asymmetric encryption and how RPass uses it for password storage.

Salting and Initialization Vectors

When saving a hash password, best practices are not to store just a hash of the password, but rather store a random value (called a "salt") in addition to a hash of the password combined with the salt.

To check if an input password is correct, the system gets the salt value used for a user, hashes the submitted password with that salt, and compares it to the stored salted hash value.

There are a couple advantages to using a salt:

- If two users have identical passwords, they'll still have different salted hashed passwords because of the randomness from the salt. If an attacker gets access to your database, they won't be able to identify identical passwords between users.

- This method also avoids precomputation using a "rainbow table" with hashed values of common passwords. A rainbow table allows an attacker to quickly ascertain if any users stored in the database use common passwords.

For symmetric encryption, there's a similar concept of an initialization vector, which is random data used to start the encryption. As with hashing, the encrypted data and the initialization vector are stored in the database.

To decrypt data, the encrypted data, the key, and the initialization vector that was used during encryption are needed. Similar to hashing, the advantages of using an initialization vector are mainly that it's harder to tell if two encrypted values came from the same plaintext value or share similar bytes in the plaintext.

RPass Security Design

The technical, nitty-gritty details of RPass's security design.

RPass security design

Security and privacy of your data is our highest priority, and our comprehensive approach reflects this.

About the Master Password and Master Key
Each user has a 256-bit master key built from the user's password and other data. This master key is created only on the user's device, and never sent to Rippling's servers. The master key is used to encrypt important data saved on Rippling's servers, so that Rippling cannot decrypt any usernames/passwords.

Because the master key never leaves the user's device, Rippling is never able to decrypt data. No Rippling employee is ever able to see a user's data, and should servers get hacked an attacker would need to brute force data encryption, which would take years with current encryption algorithms. Rippling keeps up with the latest encryption standards to stay on top of security and keep data protected with current requirements.

When you first use RPass, RPass uses the following algorithm to create a master key using your personal password (please refer to corresponding definitions):

On the User's Device:

```
1. hash1 ← PBKDF2(email, version_id, 1)
2. hash2 ← PBKDF2(password, hash1, 10000)
3. Send hash2 to Rippling's server
```

On the Rippling Server:

```
4. hash3 ← PBKDF2(hash2, server_token, 100)
5. Return hash3 to the client
```

On the User's Device:

```
6. master_key ← PBKDF2(password, hash3, 10000)
```

PBKDF2(password, salt, iterations): will represent PBKDF2 with SHA256 HMAC

Email: user's email in lowercase

Password: a user's password, normalized with NFKD

Version_id: a short, non-secret string

Server_token: a 256-bit random value stored for each user on Rippling's server. It never leaves Rippling's server — this enforces that users have to ping Rippling's server in order to convert a password into a master key

The master key requires a password and an active session (including two-factor if enabled) with the Rippling server.

Because unlocking private data requires the master key in addition to Rippling server connectivity, hackers are unable to brute-force a password offline. Any detection of a brute-force attack from Rippling servers will result in rate limits on the connection.

By rate limiting a brute force attack, an attacker is unable to efficiently run billions of possible password values in a reasonable amount of time.

Personal Vault

Each RPass user receives a personal vault.
The following steps describe how a username and password are encrypted in a personal RPass vault:

On the User's Device:

1. *data ← '{"username":"AzureDiamond", "password":"hunter2"}'*
2. *iv ← rand_bits(96)*
3. *item_key ← new AES256 key*
4. *encrypted_data ← AES256GCM_encrypt(data, item_key, iv)*
5. *item_key_iv ← rand_bits(96)*
6. *encrypted_item_key ← AES256GCM_encrypt(item_key, master_key, item_key_iv)*

These fields are then stored on the server:

(encrypted_data, iv, encrypted_item_key, item_key_iv)

Metadata fields are stored unencrypted, such as a GUID item id, created_at, updated_at, url, encryption_version, etc.

rand_bits: generates random bits using a cryptographic pseudo random number generator

AES256GCM_encrypt(data, key, iv): encrypts the data with the given key and initialization vector or nonce using AES256 in Galois/Counter Mode

Company Vault

Company vault items are encrypted with a 256-bit company key that is unique to each company. Like the master key, the company key is never stored in plaintext on Rippling's servers.

Sharing a Company Key: In order for all employees in a company to have the same company key, they need to pass the company key to new hires. The following steps explain this process.

When a new hire opens RPass for the first time, their client does the following:

1. (private_key, public_key) ← new 2048-bit RSA key pair
2. iv ← rand_bits(96)
3. encrypted_private_key ← AES256GCM_encrypt(private_key, master_key, iv)

The new hire's public and encrypted private keys (public_key, encrypted_private_key, iv) are stored on the server.

When another user within the same company opens RPass, the other user saves a copy of the company key that is encrypted specifically for the new hire:

1. *encrypted_company_key_for_new_hire ← RSA_OAEP_encrypt(company_key, public_key)*

The employee then saves (encrypted_company_key_for_new_hire) on the server.

The next time the new hire opens RPass, the stored data retrieved from other employees is passed to her along with the company key. The process is as follows:

1. *private_key ← AES256GCM_decrypt(encrypted_private_key, master_key, iv)*
2. *company_key ← RSA_OAEP_decrypt(encrypted_company_key_for_new_hire, private_key)*

Company Vault Encryption: The encryption of company vault items then functions identically to the personal vault item encryption above, except using company_key in place of master_key.

Upgrading Encryption Options
As computing power improves, encryption algorithms are no longer safe from brute-force attacks. RPass uses a system that allows it to upgrade to the latest encryption standards as newer cryptography technology is released.

Each encrypted item has an encryption_version ID stored with it, so encryption algorithms can be upgraded as needed while still being able to decrypt older items. Any new items will always be saved with the updated encryption version. RPass also has a mechanism to force older data to be re-saved with newer encryption when the user accesses it.

Server Communication
Ripping uses double encryption to save and transmit data. Data is never transferred in plain text. Data is encrypted on a user's device, and then TLS encryption is used to transfer data from a client device to a Rippling server. This double layer of encryption decreases the risk that a man-in-the-middle can read protected data.

Two-Factor Authentication
Multi-factor authentication (often called two-factor authentication) requires multiple independent credentials to successfully log into a system. RPass allows admins to require two-factor authentication for their employees to access their RPass vaults.

Unlike most other password managers, building the master key requires access to the Rippling server which enforces two-factor authentication. Thus even if an attacker has a user's password and access to the user's encrypted data, the attacker still can't decrypt the data without also having the second authentication factor.

Use of Web Crypto
RPass uses the Web Crypto API for implementations of the above encryption algorithms. Web Crypto is up to ten times faster than JavaScript implementations used in other password manager browser extensions. Web Crypto allows RPass to get cryptographically strong random values from the underlying operating system, and can also take advantage of hardware support that provides an even greater level of security.

Terminology (Appendix A)

Items: Passwords stored in RPass are often called "items" or "password items."

Personal Vault: Each user has a Personal Vault containing items that can't be shared with anyone else.

Company Vault: Each company has a Company Vault containing items that can be viewed and managed by the company's admins, and can be shared with other users in the same company.

Master Key: Each user has a 256-bit symmetric key that is basically a hash of their master password. The master key is never sent to Rippling's servers. It's generally used as a key for AES256 encryption/decryption.

Company Key (also called Broadcast Key): Each company has a 256-bit symmetric key that is used to encrypt passwords that are shared across the company. The company keys are never saved unencrypted on Rippling's servers, but they are saved in encrypted form. Again it's generally used as a key for AES256 encryption/decryption**.**

User Keys: Each user generates an asymmetric RSA public/private key pair. The public key is stored unencrypted on Rippling's servers, but the private key is only stored encrypted with the user's master key.