Tresorit Encryption Whitepaper

tresorit

Introduction

At Tresorit, we believe that every business should be given the right to take back control of their digital valuables. This vision has given birth to an end-to-end encrypted productivity solution for ultra-secure collaboration trusted by users worldwide.

Fast-paced digitalization, the accelerating shift to remote work, and increasingly sophisticated cyber-attacks, especially common ransomware attacks, have accelerated the shift to secure cloud storage providers.

Nevertheless, while end-to-end encryption (or E2EE) has been around for a while, it's not always clear what organizations mean when they claim to offer end-to-end encrypted services. There are also several best practices we believe should be industry standard and yet are not offered by several services.

Since its inception, Tresorit has been committed to offering the highest security and data protection standards for sensitive information based on client-side encryption coupled with zero-knowledge authentication and cryptographic key management. Together these form a truly end-to-end encrypted service.

Our secure productivity solution stands out due to the multi-level encryption and security concepts we use at every stage of user interaction, folder and file management, and admin operations to protect against unauthorized data access, theft or data leaks while maintaining full confidentiality of communication.

The backbone of this concept is our award-winning encryption technology that guarantees any data stored and shared in Tresorit is only accessible to you and the intended recipient(s) you have explicitly shared it with. As a result, users simply need not worry about their files in the cloud, as no data ever leaves their devices in unencrypted form.

From registration to folder or file activity, and admin supervision, various encryption and security measures ensure that your data is safeguarded according to the highest industry standards. The Tresorit Security Whitepaper is a collection of the technical background of these methods and how they work at different stages of the user journey.

What is end-to-end encryption?

Tresorit defines end-to-end encryption as a method of secure communication that ensures all encrypted information remains encrypted once it leaves the sender's device and remains encrypted until it reaches the recipient. This means that no third party has any way of accessing the exchanged information.

While encryption and end-to-end encryption are now widely adopted, many solutions provide only partial encryption, poor key management, or both. Such solutions may allow service providers to access the data stored on their servers. In even worse cases, nefarious third parties can exploit security flaws to gain access to confidential data.

To help users determine if a solution employs the highest standards of end-to-end encryption, we have collected the key characteristics of an ideal end-to-end encrypted solution:

1. **Architectural criteria:** Information exchange happens between parties through a method in which the origin encrypts the information, and only the intended destination(s) decrypt the information without any intermediary decryption.

2. **Extended definition of parties:** The parties can either be individuals or organizations. In the latter case, party refers to a system or components that are fully trusted and stay under the control of the respective organization.

3. **Key exchange criteria:** Exchanging encryption keys is done in a way that only the communicating parties gain access to the keys, ensuring no third party can access them.

4. **Key generation and management criteria:** The keys used for encrypting information are generated by the sending party, and are managed by the participating parties in a way that no third party can gain access to them, not even temporarily.

5. **Key backup criteria:** The private keys of all parties are stored in a way that no one else can gain access to them (e.g., in an encrypted format).

6. **End-point authentication criteria:** All parties can be sure that the public keys belong to the desired party, and a potential attacker cannot inject their own public key to execute a man-in-the-middle attack.

7. **Binary authenticity criteria:** The parties can verify that they are running backdoor-free software from a trusted vendor.

Meeting all the above criteria without getting in the way of day-to-day operations is a daunting task in a business setting. The goal of this whitepaper is to offer a basic overview of how Tresorit works to ensure that we fulfill the criteria above to the highest possible standards.

While our combination of end-to-end encryption and key management mechanisms is enough to offer an elevated level of security, we also employ additional industry-standard protection measures such as:

- **Channel encryption:** all communication between Tresorit's clients and servers is done on a TLS 1.2+ channel. This protects all communications delivered via Tresorit against eavesdropping and data tampering.
- **Server-side authentication** ensures that all clients are properly authenticated and only permitted to access or modify information that they are authorized to.
- **At-rest encryption** means that data is encrypted and stored server-side ensuring that the physical data and its corresponding encryption keys are stored separately.

How does Tresorit's multi-level encryption concept keep data secure?

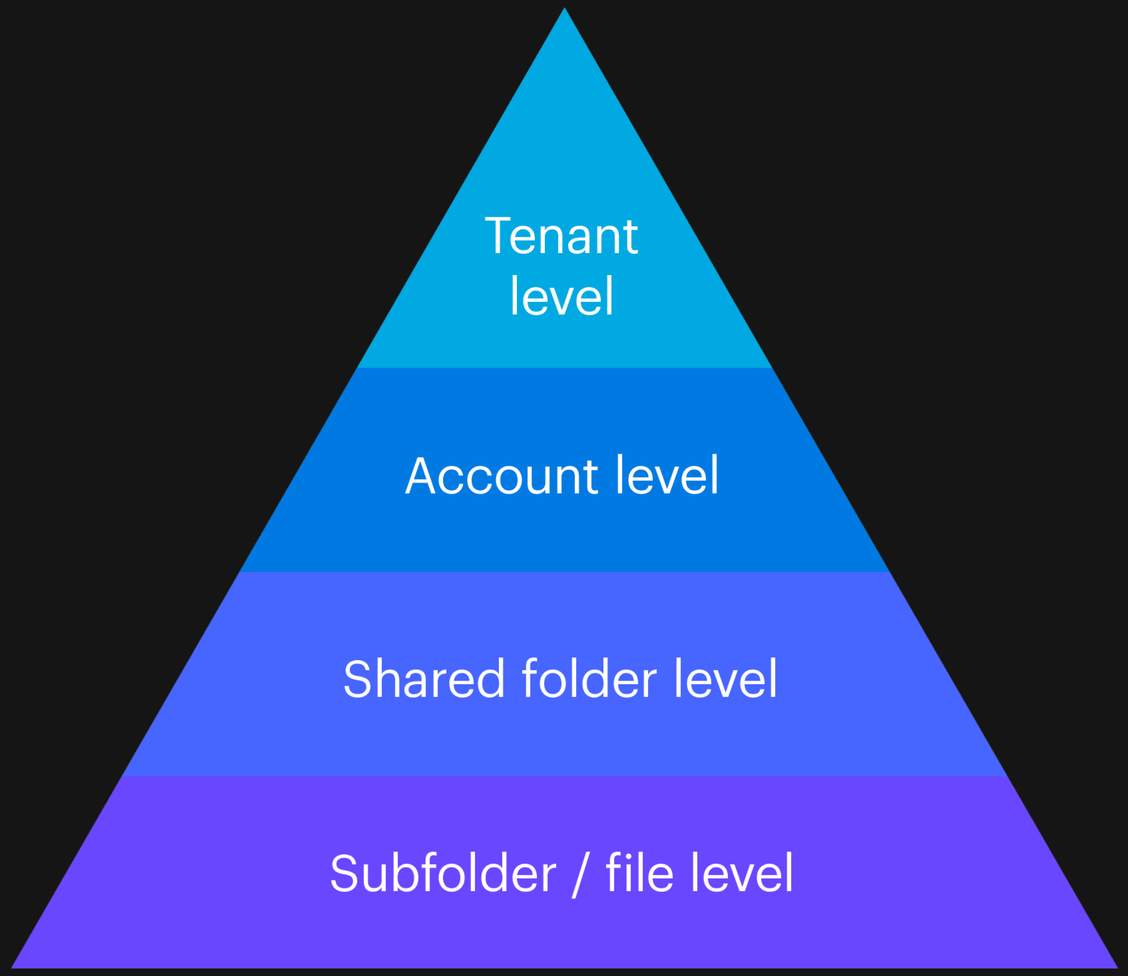

Figure 1 – Tresorit's multi-level security practices ensure that the highest cryptographic standards are applied on each level of user interactions.

This chapter will offer an overview of the cryptographic aspects of Tresorit's security. To demonstrate the multilayered concept we apply to achieve strong encryption, we will walk you through the main steps of the user journey and the encryption techniques employed in each phase:

1. **Encryption on a tenant (or subscription) level:** We provide business administrators additional capabilities allowing them to access the content of managed users, reset their passwords, control, and generate reports on Tresorit usage at a company level.

2. **Encryption concepts on an account level:** During the registration process, we ensure that the owner of the new account is identified by a verified email address and generate keys corresponding to the new user on the device used to register. These keys are the only ones that can be used to access the data.

3. **Encryption concepts on a shared folder level:** Our cryptographic mechanism and additional controls make sure that only intended recipients can access shared folders – i.e., those with whom the folders have been explicitly shared.

4. **Encryption concepts on a file level:** When uploading or accessing content, all encryption and decryption operations are completed client-side. Thus, the content of files never leaves an end-user's device in unencrypted form.

The following sub-chapters elaborate on what specific encryption methods and security controls are used for the different levels and how they work whenever operations are carried out within Tresorit.

Account level security

An account belongs to an individual user (identified by their email address) in a Tresorit tenant (or subscription).

Registration

During the registration of a new account, the user enters their email address and password along with their first and last names. The client will automatically generate a new, "empty" user profile locally. This user profile will serve as a container for future encryption keys to access the content stored by the user.

Such encryption keys are generated through a cryptographic process. A random, 256-bit password salt is generated, which together with the password is the base for deriving a 256-bit encryption key, known as the password key (derived using **scrypt** as defined in RFC 7914, with N=32, r=8, and p=1 parameters and an **HMAC step using SHA-256**). The password key is used to encrypt the user profile with **AES-256-GCM** to also provide integrity protection.

At the same time, a password validator key is calculated from the output of the scrypt algorithm with an HMAC step using SHA-256. The password validator is then sent to the server and used to authenticate the user whenever they log in from a new device.

The client will connect to the server on a TLS 1.2+ channel (to protect non-end-to-end encrypted information such as the email address). The client will send the encrypted user profile, as well as the password validator, to the server. With our zero-knowledge authentication protocol, the actual password is never sent to our servers. Thus, there is no way for Tresorit to:

- decrypt any information that was encrypted using the password of the user (as the key derivation functions used are one-way operations)
- restore the password of a user in case it was forgotten (see below for more information)

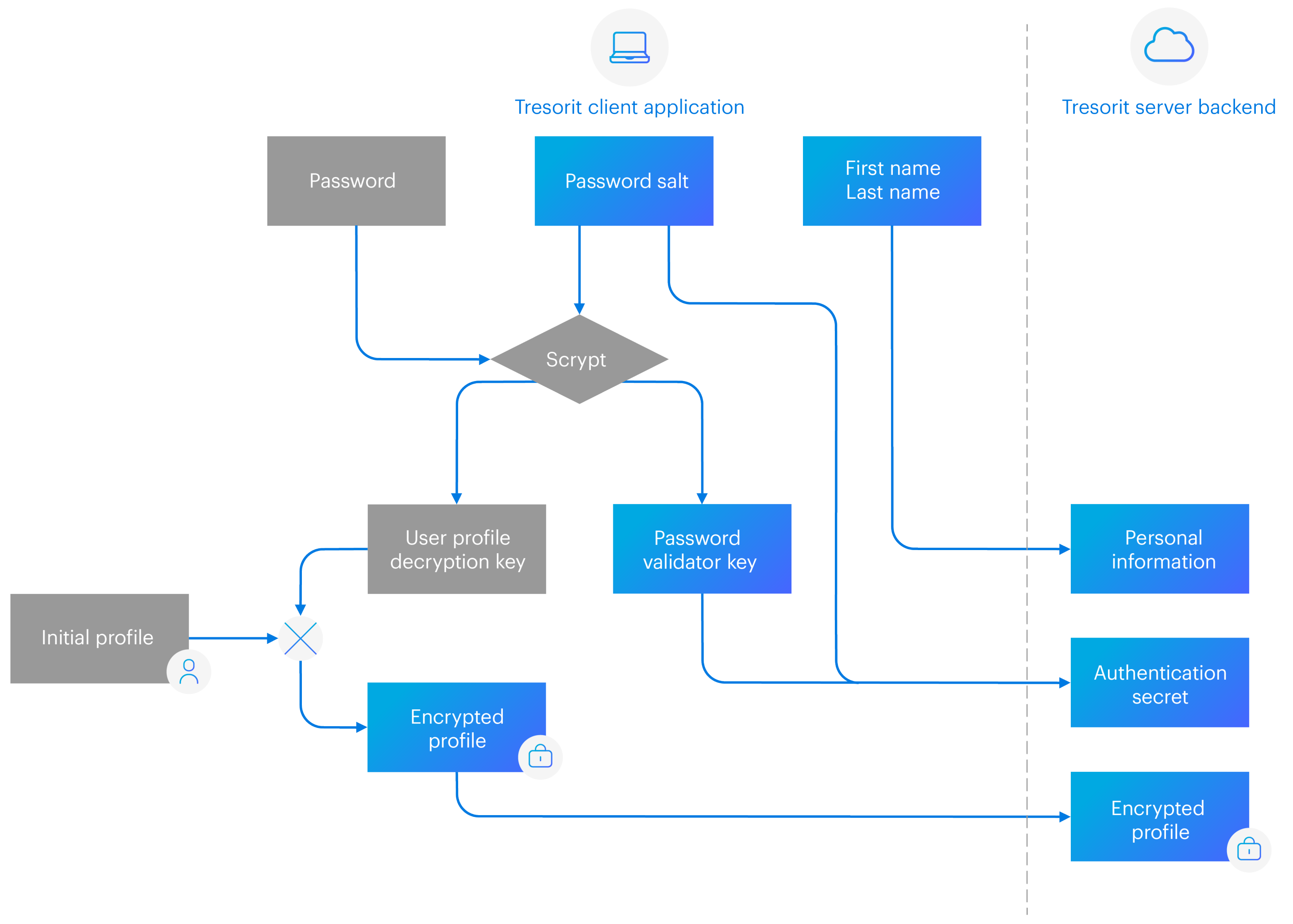

Figure 2 – Tresorit's registration process.

After the registration process, additional security features can be enabled, such as multi-factor authentication. At least two of the following methods must be turned on to ensure secure access to the account in question: email, text, voice, TOTP.

As previously mentioned, the user profile contains all encryption keys needed to access the content of the user (folders, files, shared links, file requests, etc.). This encrypted container is constructed in a way that allows multiple, independent decryption keys to access its contents. The solution is enabled by a combination of **AES-256 GCM in an AEAD scheme with RSA-2048 and OAEP padding**.

The figure below illustrates the process in the case of a password. A password key is derived using the above-mentioned scrypt algorithm. The password key is then used to decrypt an RSA private key that serves to decrypt the profile key encryption key, which can be used to access the contents of the profile itself.

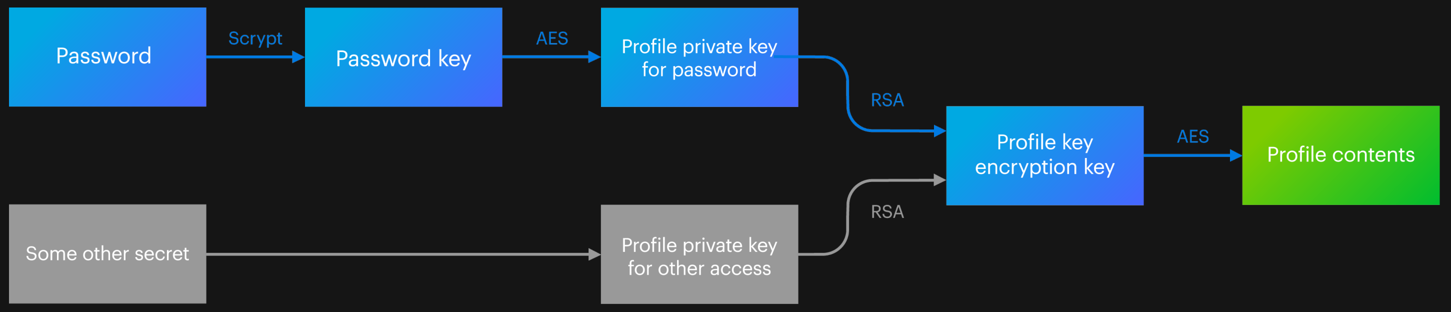
Password
Scrypt
Password key
AES
Profile private key for password
RSA
Profile key encryption key
AES
Profile contents
Some other secret
Profile private key for other access
RSA

Figure 3 – Flowchart of the password authentication process.

Login

When logging in from a new device for the first time, a user must enter their email address and password (and additionally complete multi-factor authentication if enabled). A proprietary challenge-response protocol is used for authentication based on the salt and password validator established during the signup process. Tresorit uses a PBKDFv2 and HMAC based challenge-response protocol, like CRAM-MD5 (RFC 2195). Password derivation is handled by **SHA-1 and PBKDFv2 with HMAC with SHA-1**, as recommended in NIST 800-132.

During the process, the server generates a 160-bit random nonce and sends it to the client with the salt as a challenge. The client then assembles the response value, which contains the salt received from the server, and a newly generated 160-bit client-side temporary salt value. It then calculates the password validator and derives the actual response using **PBKDFv2 with SHA-1 and 1 iteration**, which it sends back to the server. The server can also calculate the proper response (since it previously stored the password validator value), and the authentication process is finished if the two values match.

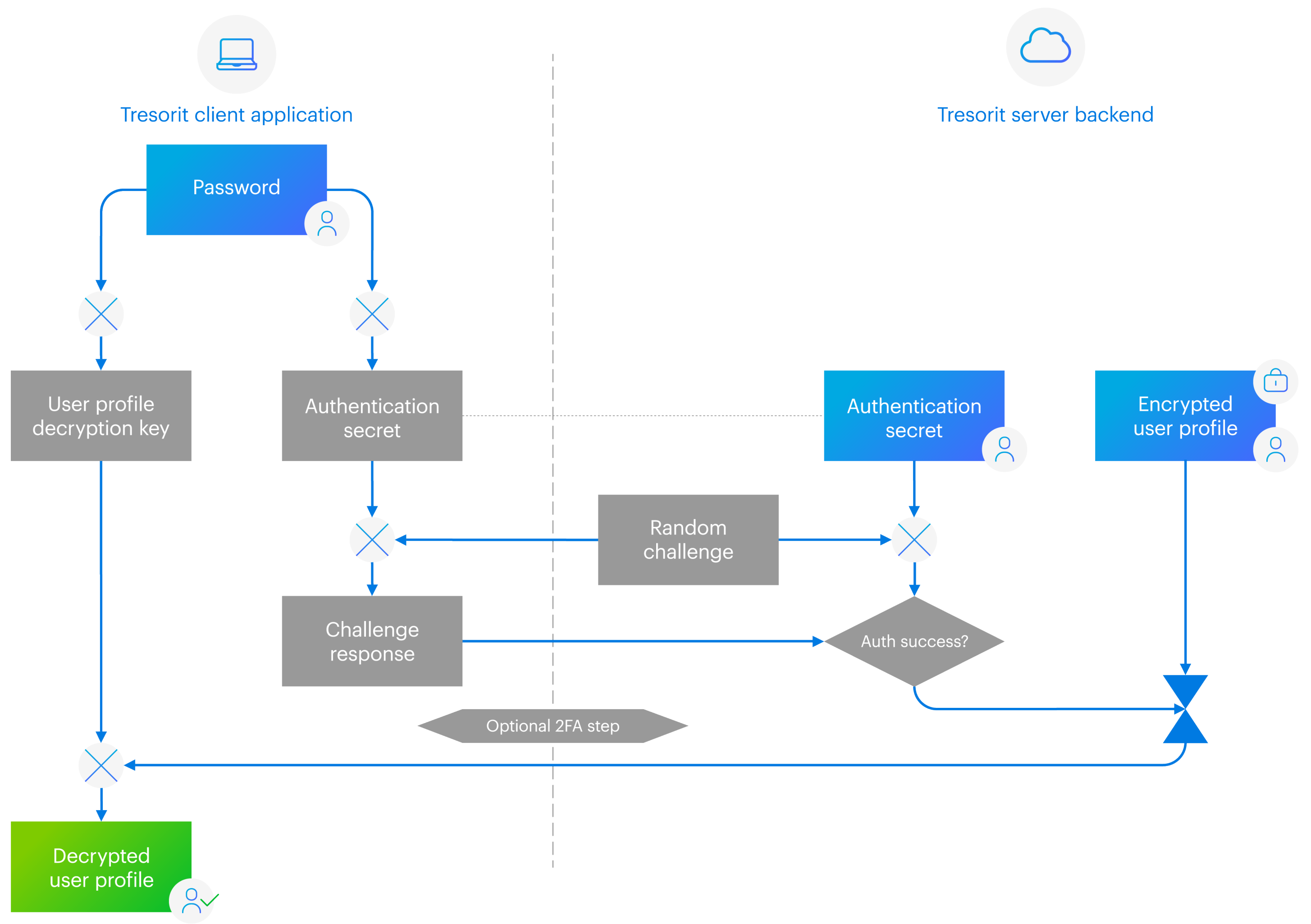

Figure 4 – Flowchart of user authentication between the Tresorit client and the server. All communication between the client and the server is perfomed on a TLS 1.2+ channel

All communication is performed over a **TLS 1.2+** channel, and the password of the user never leaves their device during the process.

After successful authentication, the encrypted profile is sent to the client. The profile is decrypted locally using the password key calculated by the client with the previously entered password.

Additional device-specific keys are generated after decryption of the profile. One of these is used to encrypt the profile key encryption key and is then stored in the platform-specific locally encrypted storage provided by the operating system. This key will be used to decrypt the profile in the future if auto-login is enabled on the corresponding device. The other key will be used for communication with the server to sign all API requests.

If auto-login is not enabled on the device, but a successful login has previously been completed using the same device, a local profile will be decrypted. A device-specific local key is used in this method, calculated from the password using the **PBKDFv2 algorithm with SHA-256** (FIPS-180-4) **with 100000 iterations**. This key is different from the key used to encrypt the user's encrypted profile.

Password change

As the password key and the password validator (both irreversibly derived from the actual password) are used both for encrypting the user profile and authenticating to Tresorit's servers, changing passwords requires a sophisticated process.

First, the client initiates the same challenge-response protocol as described in the first-time login process using the old password. Then a new password salt and authentication salt are generated, and the new password key and password validator are calculated from the new password according to the process described above. The user profile is re-encrypted locally using the new key, and the client sends the newly encrypted profile and the new authentication salt and password validator along with the response. This concludes the password change process.

Please note that both the old password and the Tresorit client application are required for the password change process. In addition, Tresorit servers also require it to prevent malevolent actors from changing a password in the unfortunate case when a device is left unattended, unlocked, with access to the Tresorit client.

The fact that the password change process involves the complete re-encryption of the user profile makes server-side password recovery theoretically impossible, which is a noteworthy technical point.

Password reset

However, the above do not mean that a user loses access to their data in the event of a forgotten password. Normally, both the password verifier and password key are required to change a password. However, these are both derived from the password itself.

Tresorit has no access to passwords as these never leave the user's devices. Therefore, the password reset feature does not recover the actual password but helps in setting up a new password securely.

If the user is already automatically signed in to the client application, a device-specific private key is available without knowledge of the password. The profile is directly decrypted with this locally stored private key during automatic sign-in. The device-specific private key never leaves the device and is protected using platform-specific security measures.

However, the user must still prove their identity to our servers. Because the password is unknown during automatic sign-in, there is no way of calculating the password validator that would be used by the challenge-response protocol during authentication. To replace it, a randomly generated, one-time authentication code is sent to the email address associated with the Tresorit account. Similarly to multi-factor authentication, we assume that if the user can obtain the authentication code, it proves that they are in control of the respective email account. Derivatively, this means they can also be in control of the associated Tresorit account.

This code will be used to initiate the password change, which takes place as described above. A new pair of keys are derived from the newly entered password. The profile is re-encrypted and then uploaded to the cloud. The process also updates the password validator.

The user's devices employ different keys to access the encrypted profile. These are not changed during a password change, and their cryptographic access and active sessions will not be revoked. This can be done separately.

Shared folder security

A shared folder (tresor) is a top-level folder that can contain several sub-folders and files and can be shared with other Tresorit users to collaborate on the contents.

To be able to access (or create) shared folders, a user must have the tresor share key. When creating a new shared folder, this public-private key pair is generated locally, and the private key is stored in the user profile. The server then saves the locally updated encrypted profile after it is uploaded by the device and associates the public tresor share key with the user. All the private tresor share keys are stored in the encrypted user profile, and their public counterparts are managed by the server. Key rotation for these is done automatically by the user device within reasonable intervals, or when a given cryptographic event (such as a device revocation) necessitates it.

When creating a shared folder, a new tresor key file is generated. This is a similar construct to the encrypted user profile. This file is the container for all further encryption keys associated with the files and subfolders and is a shareable structure. The tresor key file is encrypted with a combination of **AES-256-GCM in AEAD scheme and RSA-4096 with OAEP padding**. This encrypted tresor key file is uploaded to the server during the folder creation process alongside some metadata that is also stored on the servers.

Whenever a folder is shared, data is protected at two levels:

- **End-to-end encryption:** only those who can open the tresor key file can read any data that is stored inside a shared folder. The end-user has full control over who can access their data.
- **Server-side ACL (Access Control List):** additional permissions are handled via classical server-side controlled permission levels.

Accounts can have different access permissions within a shared folder:

1. Viewers have the lowest level of access; they can download, decrypt, and view the contents of the folder.
2. Editors can upload, modify, and delete content.
3. Managers have the right to share the folder with additional users.
4. Owners own the data of the shared folder and can modify manager roles, invite new managers, or even delete the whole folder.

During the sharing process, the user opens the tresor key file using their own tresor share private key (stored in their user profile). The recipient's tresor share public key is also retrieved from the server-side by the user's device. The tresor key file is then updated locally, shared with the recipient, and uploaded to the server where it is stored. Additional information required for bookkeeping the member list required for ACL is stored alongside this data.

Whenever access to a shared folder is revoked, a similar process happens – the tresor key file is re-encrypted so the affected user can no longer decrypt it.

Whenever a membership change occurs in a shared folder, a brand-new key is generated and is used to encrypt all future content. This ensures that no one who previously had access to the shared folder can access any newly created content (a method called lazy re-encryption). While the ACL will limit access to old data immediately when rights are revoked, this method does not require the data in the shared folder to be re-encrypted immediately.

Subfolder and file level security

Files are organized in an arbitrary structure of folders and subfolders within a shared folder (tresor). These files can either be shared individually via links or on a folder or shared folder level.

Tresorit uses a symmetric key encryption algorithm, more specifically **AES-256, in OpenPGP CFB mode** described in RFC4880 to encrypt all uploaded files and folders. Every file has a unique, independent, and freshly generated 256-bit encryption key.

Each version of a file has a random 128-bit IV. As a result, its encrypted form changes completely, even if only one bit is changed in the file. In practice, this ensures that neither Tresorit nor others have any information about the changes made. Folders are encrypted the same way, and the integrity of all ciphertexts is protected with **HMAC-SHA-512**.

The file structure created client-side is mirrored on the servers, meaning files and folders are stored in the same hierarchy. The keys and metadata of each file or folder are stored in an encrypted format in their parent folder. Each file or folder has a randomly generated, unique, 32-byte file identifier on the server-side.

A fresh file encryption key is generated by the device for each new file, or when shared folder membership has changed since the last time an existing file was uploaded (lazy re-encryption).

The parent folder is accessed (decrypted via the user profile and then the tresor key file to access the root folder). The file name, modification date, file identifier, and the file encryption key are stored inside, and the folder is re-encrypted. The server stores the new encrypted parent folder and the encrypted files as well. Some metadata is stored with the file, including the date of upload and the uploader’s email address.

Link-based sharing

Tresorit offers users the option to share files and folders with others who may not have a Tresorit account or when there is a one-time file transfer that makes creating and sharing a folder unnecessary. As a result, technical solutions must be put in place to ensure data security when a user cannot be authenticated with their email address and password-derivative authentication information, and it’s essential to ensure the provider has no access to data. When files are shared with links, the sharing link itself contains the data needed to access the information stored within.

When a user creates a link to a shared file or folder, the Tresorit client application will generate a link in the following format:

https://web.tresorit.com/l/<pathId>#<clientSecret>

Both <pathId> and <clientSecret> are freshly formulated random identifiers:

1. <pathId> is a 5-character random identifier that serves only server-side validation and load-balancing. It is not involved in the encryption of the content.

2. <clientSecret> is a 128-bit random sequence encoded as **Base64** (RFC 4648) that carries the access information for the shared content. The secret is generated on the client-side and placed into the fragment part of the URL. It is never transmitted to the Tresorit backend, neither by the client applications nor by the browser used to open the shared link. Meaning Tresorit cannot use it to access the shared data.

When a shared link is opened, the client-side application running in the browser will extract <clientSecret> from the URL and uses the key derivation function **PBKDFv2** to calculate two 256-bit derivative values: <linkId> and <encKey>. Note that PBKDFv2 is a one-way function so it is impossible to recreate <clientSecret>, or <encKey> from only the <linkId>.

These values are used as follows:

1. Using <linkId> as an identifier, the application obtains an encrypted metadata package from the Tresorit backend, which was stored there when the link was created.

2. This package is then decrypted using <encKey> or, in the case of password-protected links, the combination of <encKey> and the link password. In practice, there are additional cryptographic steps involved here to enable additional security measures like the option to show the name of the shared file before asking for the password and to prevent brute-force attacks on the password.

The decrypted package contains all information needed to access the actual shared content, including:

- the name of the shared file
- the server-side identifier of the encrypted file
- the 256-bit file decryption key

The client application then downloads the shared content, decrypts it locally, and presents it to the user.

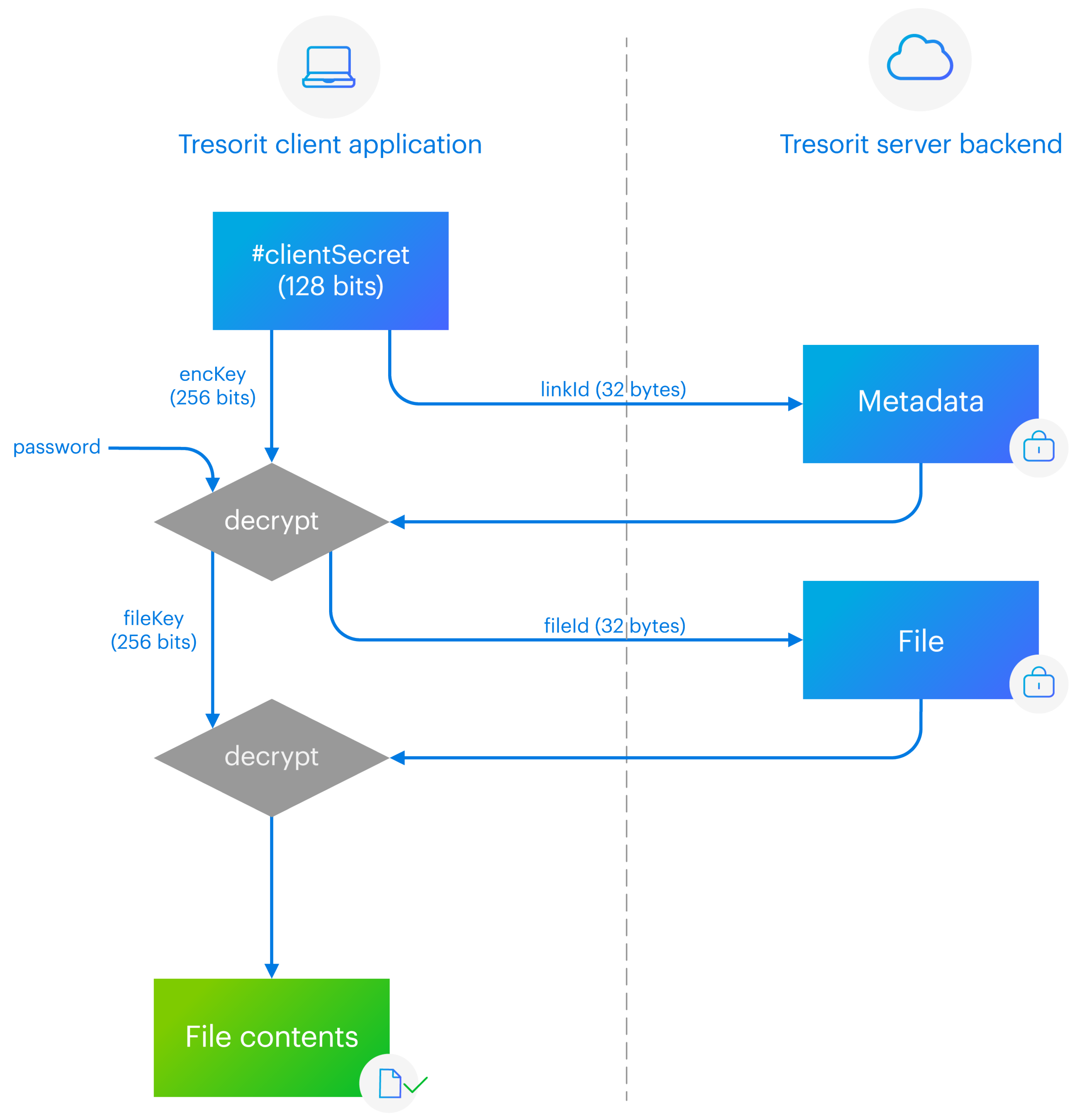

Figure 5 – Simplified flowchart showing the process of opening a file using a shared link

Advanced security controls for shared links

Tresorit provides users a way to share files securely inside and outside their organization via end-to-end-encrypted links. Thanks to the wide range of security settings that can be customized based on the recipients' profile and share intent, users can keep full control over the content they've shared:

- managing **link security with passwords, expiration date and open limits**
- **allowed viewers**: allowing access to shared content for specific email addresses or domains
- enabling **access logs** to track shared files and see the email address, open date, IP address and platform used to open the shared links
- **advanced link tracking** to keep an eye on the number of times a file was viewed, downloaded or saved to a Tresorit account
- **document analytics** to get insight how recipients are engaging with the shared document, down to the page level
- avoiding unauthorized re-sharing of documents with **dynamic watermarks**
- **disabling download/print**

Tenant level security

Countless Tresorit users are employees of organizations that have selected Tresorit as their secure cloud storage and file-sharing provider. However, as all information is end-to-end encrypted based on user identifiers, admins could run the risk of losing control over company files and access to those stored by their employees, should a user leave the tenant organization for whatever reason, from maternity leave, resignation, or even a sudden tragedy. In other cases, an organization can even be legally required to share files stored by their employees with the authorities or oversight bodies. This is in the hands of your organization and user administrator – Tresorit cannot access and hand over any customer data to regulatory bodies.

To avoid these risks and aid compliance, Tresorit has set up robust controls to ensure admins never lose sight of and always have access to data stored within their tenant organization. These systems allow Tresorit to offer the highest standards of security while managing all the risks connected to end-to-end encryption.

Advanced control

Advanced control gives tenant administrators – and only them – the ability to reset user passwords or completely take over user accounts within the tenant. It relies on the flexibility of user profiles, enabled by setting up the tenant-wide administrative recovery key as an additional decryption method.

When a new user is registered to Tresorit, they accept their tenant administrator and automatically encrypt their profile with the public key associated with the tenant. This later enables the subscription owner to reset their password when needed.

The tenant administrator can initiate the password reset when their administrative recovery private key is accessed from a secret kept in their user profile. The administrative recovery private key will be stored in local memory during the entire process. The encrypted user profile of the target user is retrieved from the server and is decrypted locally with the administrative recovery private key.

An encryption key is then derived from a newly generated password and used to re-encrypt the target user profile. The encrypted user profile of the target user is uploaded to the server along with the password validator that corresponds to the newly generated password. The target user can use the newly generated password to log in to their account and change their password.

Single Sign-On

Single sign-on (SSO) integration allows selected users of the tenant to use their company-wide Azure AD (Active Directory) or Okta credentials to sign in to their Tresorit account instead of a dedicated password. When using SSO authentication, the end-user does not have a Tresorit password. Both the server-side authentication and client-side decryption must be based on other components.

When signing in to Tresorit via SSO, users enter their (company account) password on the SSO provider's own login screen to which the Tresorit application does not have access. The SSO provider will return a claim to the client application that proves that the user has successfully authenticated themselves.

Based on the trust relationship set up when admins configure SSO for Tresorit subscription, Tresorit's backend will be able to verify the origin of the claim and provide access to the encrypted user profile. The Tresorit client application can use the claim to obtain a second secret from the SSO provider to decrypt the contents on the client-side.

Tresorit relies on the SSO provider's claim to decide whether to allow access to the encrypted data. The user profile is eventually decrypted with a key stored at the SSO provider.

Whether login happens using standard password-based authentication or SSO, Tresorit, as a service provider, never receives your password or gets access to any of your encrypted content in any intelligible, un-encrypted, or decrypted form. All cryptographic operations are done on the client-side to maintain end-to-end encryption and zero-knowledge.

When admins set up SSO for a Tresorit subscription, they are trusting the SSO provider to access and store Tresorit's user profile encryption data that previously existed only in the memory of their workforce. This information is shared with the SSO provider in addition to the information for verifying passwords that the SSO provider already has. In most cases, this is not a new or critical concern, as all other company systems that process sensitive information (email, device access, additional cloud storage, payroll CRM, etc.) already have SSO configured. Should such a setup exist, this risk has already been assessed and accepted.

However, Tresorit applies additional security measures to ensure SSO is used in a secure and flexible way while satisfying companies' specific needs:

- Administrative actions requiring Advanced Control (for example, resetting subscription members' passwords) will still require the subscription administrator's Tresorit password, even if SSO is enabled for the administrator.

- By using separate policy templates, admins will have detailed control over who should use SSO and who remains on password-based authentication. Thus, admins can create a circle of employees who manage data of the highest sensitivity with the highest security Tresorit has to offer while allowing others the convenience of SSO.

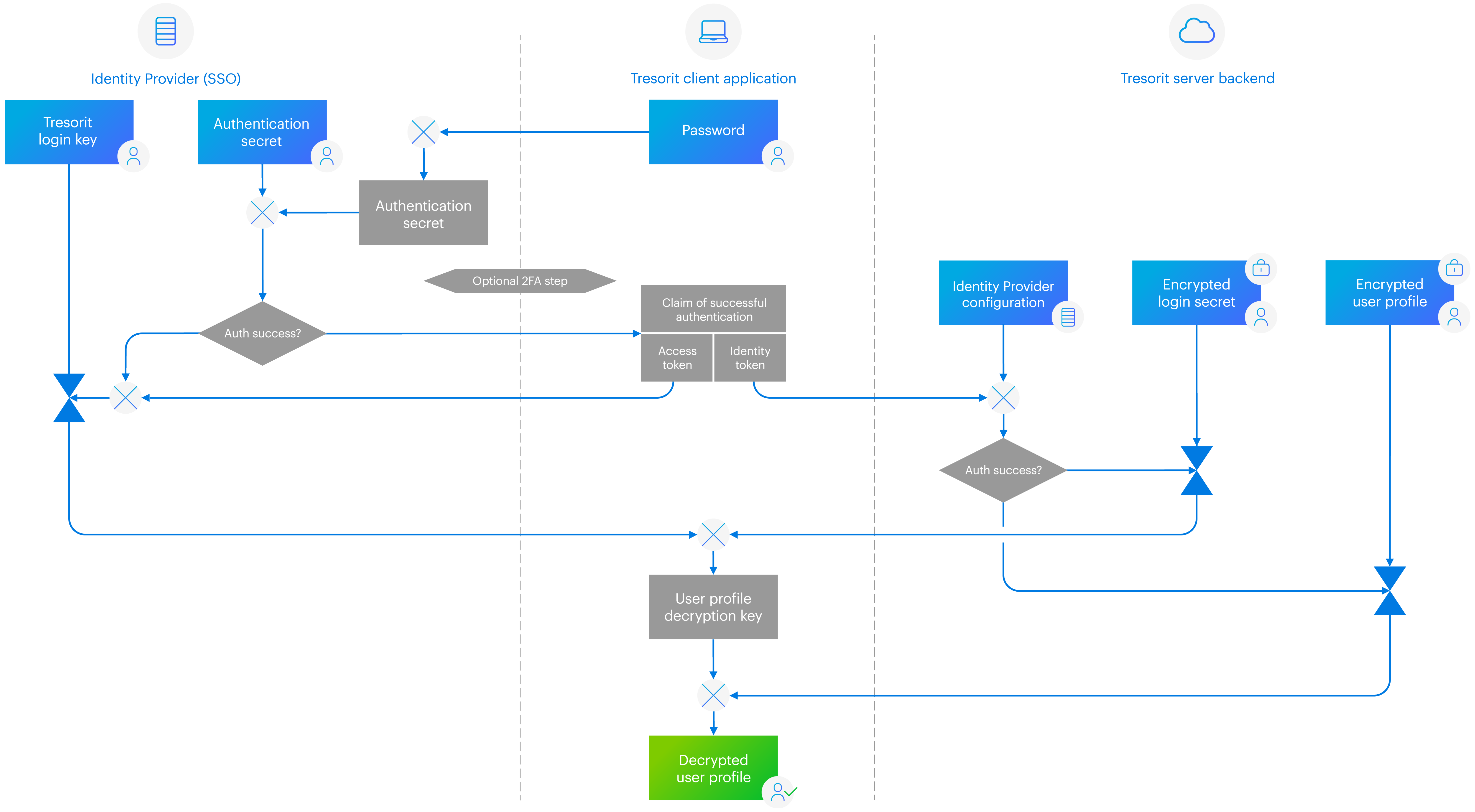

Figure 6 – When SSO is used, the identity provider sends the Tresorit client a claim that the user has authenticated themselves successfully. This is used to obtain a second secret to decrypt the user profile.

Additional tenant level controls

Beyond cryptographic access, administrators (and co-administrators) have several controls that can be enforced on the server-level to further enhance the security level of an organization and aid them in staying in control of their data. By setting up custom policies, they have the capability to restrict how users can use the application and different functions, such as:

- enforcing **2-step verification**,
- **defining allowed devices** and platforms to limit the operating systems and mobile platforms certain user groups can use to access Tresorit,
- setting **IP filtering and session control** to control the IP addresses from which selected user groups can log in to Tresorit,
- **disabling remember-me**,
- **disabling permanent deletion**,
- **data residency options** that allow admins to specify the location of data centers to satisfy company policy or industry and jurisdiction-specific regulations,
- **disabling tresor creation, sharing and synchronization** to support the creation of group policies that prohibit or limit sharing rights.

What are Tresorit client applications?

Users can access the Tresorit service with a client application installed on the user's device, or alternatively, the web-based client. The client application handles all **encryption**, **decryption**, **key management**, and **sharing**, **file synchronizing logic** on the **client-side**. To ensure binary authenticity, client applications are signed by Tresorit. This signature is checked automatically before any update happens. The application is available for all major desktop and mobile platforms:

- Windows (7, 8, 8.1, 10)
- Mac (OSX 10.9 or later)
- Linux (Ubuntu 16.04+, and all Linux distributions with a window manager from the last 4 years)
- Android (5 or later)
- iOS (11 or later)

The web-based Tresorit client requires JavaScript, and can be accessed from the following browsers:

- Chrome 60 or later,
- Firefox 60 (ESR) or later,
- Safari 11 or later,
- Edge 77 or later
- Brave

JavaScript is needed because the application runs inside of the browser and completes all encryption and decryption operations locally. The same principles apply as with all Tresorit instances:

- All files, folders, profiles, etc., are encrypted within the browser's memory before they are uploaded.
- All files, folders, profiles, etc., are downloaded in encrypted form and decrypted within the browser's memory before being displayed to the end-user.

Conclusion

We at Tresorit believe that to ensure data security against the threats of the digital world, solutions must not only offer protection but also be easy to use. If users circumvent secure solutions because they are difficult to set up, deploy, or get in the way of everyday tasks, they remain unprotected.

That is why the multi-level encryption technologies detailed above are constantly at work in the background to ensure data remains secure while providing an outstanding user experience. Despite our technology being centered around our zero-knowledge end-to-end encryption methodology, several other security features ensure the confidentiality and safety of user communication inside and outside of user organizations.

These security features have been designed to satisfy the strictest data protection and compliance requirements of highly regulated industries such as financial services, healthcare, legal, manufacturing and even defense contractors. This unique combination of security features coupled with our data residency options support organizations in staying compliant with local and industry-specific legislations, such the GDPR, FINRA, HIPAA, TISAX, and CCPA.

Today, organizations are faced with complex challenges when it comes to digital security and compliance. That's why Tresorit is committed to understanding their needs and creating a cloud productivity solution to protect sensitive data and support collaboration in offices, fully remote workforces, and across borders.

Learn more about our various encryption technologies, security, and compliance features at tresorit.com, and make sure to follow us on LinkedIn and Twitter to stay up to date with the latest news about our solutions.