zivver

Zivver's security and privacy by design approach explained

Introduction

Introduction

Most people think of encryption when talking about secure communication. They think of hackers being their primary risk. However, in practice, most data leaks are caused by human error, such as sending information to the wrong person.

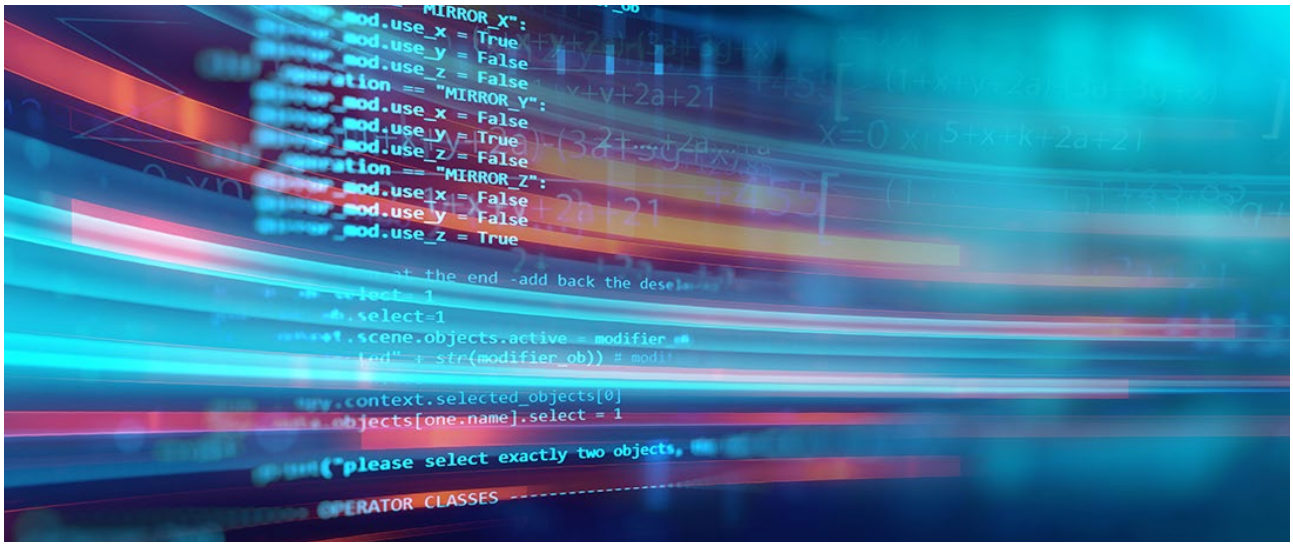
So, looking at encryption alone is a very limited way of considering secure communication. It only manages the 'during' sending risk, while leaving the issue of human error before and after sending untouched.

Although encryption is not the panacea, it is an important line of defense. However, it should be used in the correct way and knowledge about the topic is often limited.

In this paper, we outline how Zivver uses encryption to ensure that only the appropriate sender and recipients can read the secure messages sent via our platform.

A quick Google search defines encryption as follows: "Encryption is the process of encoding a message or information in such a way that only authorized parties can access it. Encryption does not prevent interference, but denies the content to a would-be interceptor." This is exactly what we do. We ensure that the data of our users and their contacts is safe. In effect, this means that sensitive data should also be inaccessible to Zivver - meaning we are neither a possible source of data leaks, nor an attractive target for hackers.

We don't have possession of the keys needed to decrypt information. This requires the use of encryption algorithms reliant on separate keys for encryption (public keys) and decryption (private keys), also known as asymmetric encryption. However, strong encryption requires the use of strong keys – in other words, a very long password.



We empower users and organizations to protect their sensitive digital communications, effortlessly.

For those who take data security seriously, this poses a challenge: the provider cannot store the key, but the user requires it in order to decrypt the data. So, what is a practical way of managing keys, while remaining both user friendly and secure? This is something 'traditional' encryption methods, such as PGP, failed to accomplish, as it relies on users storing their 2048 bits key somewhere safe.

In order to deal with this challenge, we found an innovative solution.

In this paper, we outline the principles and functionality of this solution in all its facets, covering:

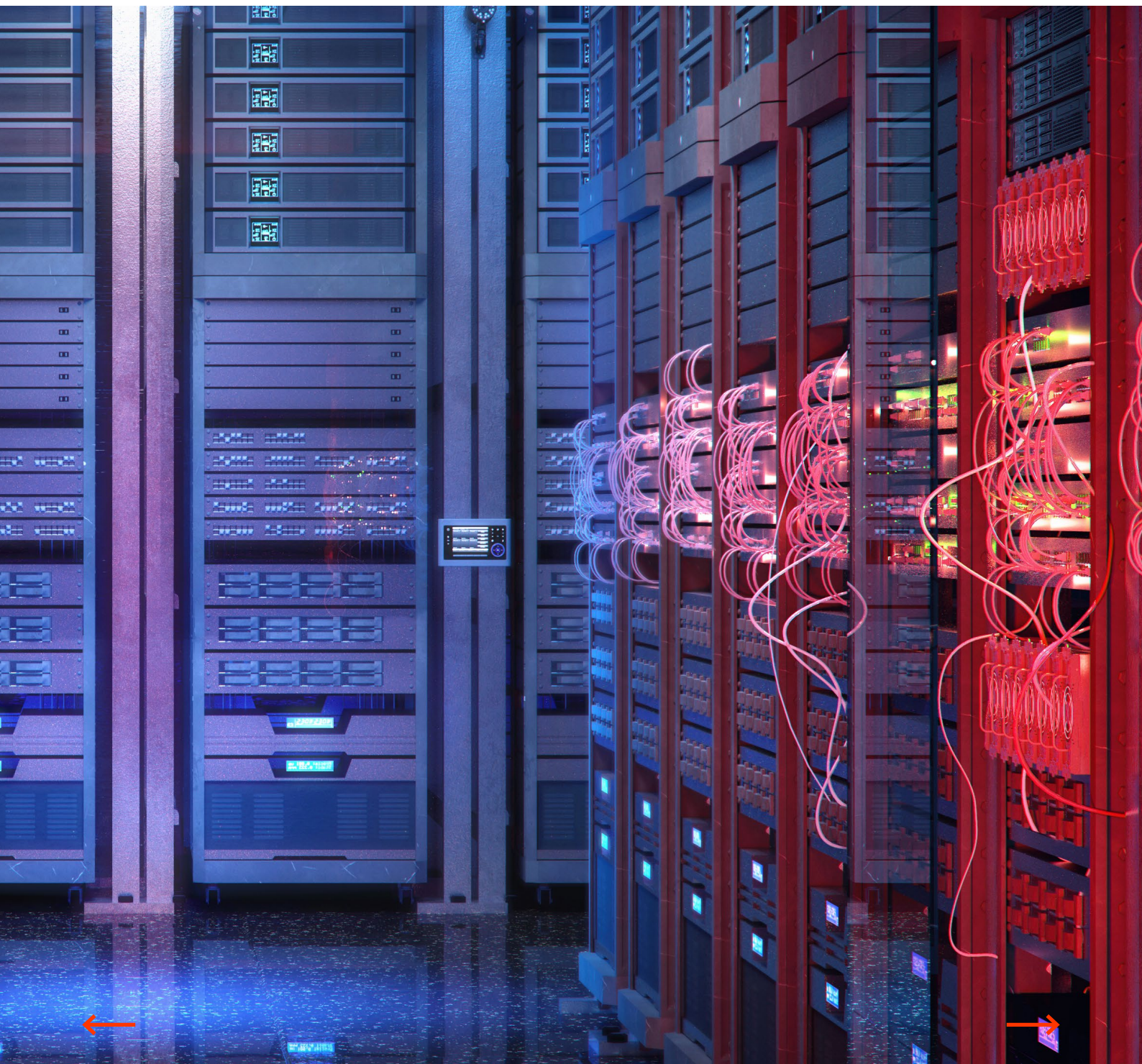
1. Transport Layer Security
2. User authentication and log-in
3. Public/private key creation and management for new users
4. Information sharing with other users
5. Information retrieval by users
6. Information sharing with guests
7. Information retrieval by guests
8. Master key options for organizations

All communication between clients and the Zivver platform is done via an API (Application Programming Interface) based on REST-principles, using JSON to transmit data objects. In this whitepaper we will sometimes refer to API-calls to clearly explain how the platform functions.

1. Transport Layer Security

For all communication between clients and servers, Zivver uses TLS 1.2 and TLS 1.3, depending on the capabilities of the user (RFC 5246).

The TLS (Transport Layer Security) protocol provides privacy and data integrity between two communicating applications. It is the most widely deployed security protocol today, and can be used by web browsers and other applications that require data to be securely exchanged over a network.



2. User authentication and log-in

For user authentication and granting access to accounts, Zivver relies on a combination of the user providing valid credentials in the form of a user name and a 'secret', usually referred to as a password, and a second authentication factor.

2.1 Credential verification

For a user with a Zivver account, credentials need to be provided upon account creation. For Zivver, the user's (primary) email address is the username. For the secret or password, Zivver provides the following options:

- A password chosen by the user. This includes users of personal accounts, for administrators, or for users of organizations that do not use Single Sign On (SSO) to log into Zivver.
- A secret provided by a SAML 2.0 or OICD compliant identity provider (IdP) during the login-call. This usually holds for users of organizations that have Single Sign On e.g. via ADFS setup.

Zivver technically allows logging in with both a password and one or more secrets provided by SAML 2.0 or OAuth 2.0 clients. For business users, however, the organization often chooses to only support logging in via their SSO-IdP.

Passwords are stored using BCrypt. Zivver uses BCrypt, which has a per user unique salt generated with a cryptographically secure pseudo-random number generator (CSPRNG), to store hashes of users' secrets.

2.2 Second factor authentication

Zivver only allows users to send messages if their account is protected with another layer of security, also known as second factor authentication. For users with a non-business account, Zivver supports logging in with an additional SMS-code (a TOTP-based code sent by Zivver, via an SMS-provider) or via the use of an authenticator app compatible with the TOTP-standard (RFC6238).

Users are required to set up this second factor when logging in to Zivver for the first time, or every other time the user tries to compose or reply to a message. Part of the second factor setup is the (strongly encouraged) possibility to download ten, one-time use, backup codes as a .txt file. They can be used in case the user does not have access to the second factor (usually a phone).

For business users, Zivver allows logging in via a SAML-v2 authentication context. As described in section 2.1 Credential Verification, Zivver allows organizations to use SSO via SAML 2.0 to authenticate their users in Zivver. In addition to username and a secret (password), Zivver allows organizations to pass an authentication context in the SAML response indicating whether a user was authenticated and in which way.

This means that organizations do not require users to log in to Zivver with a second factor in case the IdP of the organization already required the user to log in with a second factor. In that case Zivver acknowledges the IdP's additional authentication context. Zivver does not challenge the user for a second factor itself. In this scenario, we help the user by not bothering him twice without losing any aspect of security

Users are always challenged to provide proof of possession of their account-configured second factors. They have to provide this proof once they log in to Zivver with valid credentials on an untrusted device. When entering their second factor, Zivver clients can allow users to select an option to trust the specific device. If the user chooses to do so, Zivver issues a unique device-specific key that should be provided by the API client with every login-call as a proof of being a trusted device.

For business users Zivver allows logging in via a SAML-v2 authentication context. As described in section 2.1 Credential Verification, Zivver allows organizations to use SSO via SAML 2.0 to authenticate their users in Zivver. In addition to username and secret (password), Zivver allows organizations to pass an authentication context in the SAML response indicating whether a user was authenticated and in which way. This allows organizations to avoid that users need to log in to Zivver with a second factor in case the IdP of the organization already required the user to log in with a second factor. In that case Zivver respects and acknowledges the IdP's additional authentication context. Zivver does not challenge the user for a second factor itself. In this scenario we help the user by not bothering him twice without losing any aspect of security.

Users are always challenged to provide proof of possession of their account-configured second factors. They have to provide this proof once they log in to Zivver with valid credentials on an untrusted device. When entering their second factor, Zivver-clients can allow users to select an option to trust the specific device. If the user chooses to do so, Zivver issues a unique device specific key, that should be provided by the API-client with every login-call as a proof of being a trusted device.

2.3 User login and future authentication with access and refresh tokens

After a successful login call, following the OAuth 2.0 specification (RFC6749), the API returns both an access and a session token. The API client must use the access token to access API endpoints that require authentication.

The refresh token is valid for a set period of time based on the client, and can only be renewed by the user specifying the password. A session can be terminated (i.e. logged out) at any moment by presenting either the access or token to the revoke endpoint (RFC7009). Existing session tokens are invalidated server side, because the token lifetime cannot be changed in retrospect after being issued to the client.

An access token has a short lifetime of just minutes. If the access token is expired, the refresh token must be used to obtain a renewed access token.

3. Public/private key creation and management for new users

The basis of the Zivver platform is formed by the use of a public-key cryptosystem in combination with some other methodologies, to align as closely as possible with the zero knowledge objective. Whenever a new user account is created, the following process is followed:

3.1 A new user account is created by providing username and password

Either the user or the organization creates a new account by providing a username and password as a part of the corresponding API-calls. We also support SCIM 2.0 for account provisioning.

3.2 Create new public/private key pair

For every new user that creates a Zivver account, a new public/private key pair is generated using a CSPRNG. Encryption is performed using the RSA-algorithm (Rivest–Shamir–Adleman, 2048 bits).

3.3 Store public key of user

The user's public key is stored in the Zivver platform for later use to encrypt files and messages.

3.4 Generate derived key based on user password

The user's password, in combination with a unique 'salt', is used to generate a hash using PBKDF2 (Password-Based Key Derivation Function 2). The PBKDF2 key derivation function looks like Derived Key = PBKDF2(PRF, Password, Salt, c, dkLen), where PRF is a pseudorandom function of two parameters with output length hLen (e.g. a keyed HMAC),

i Password is the user's password from which a derived key is generated

ii Salt is a sequence of bits, known as a cryptographic salt

iii c is the number of iterations desired, 65,536 in the case of Zivver

iv dkLen is the desired length of the derived key, 128 bits in the case of Zivver

3.5 Encrypt user's private key

We encrypt the generated private key of the user using the AES-CTR algorithm (Advanced Encryption System, with Counter mode, 128 bits), using the derived key from step 3.4 as a symmetric encryption key.

3.6 Store encrypted private key

We save the user's encrypted private key in our database.

3.7 Clean-up unnecessary data

Finally, we remove all (sensitive) information that isn't required from memory, including the user's password and private key.

The above steps are conducted fully in-memory in (any of the) Zivver servers. Therefore, neither the user's password nor private keys are ever stored. It is never accessible for anyone while 'at rest'.

An analogy of this process could be considered as follows: Together with the user, we have prepared a meal and written down the full recipe. However, we cannot prepare the recipe again without he user's presence, as the secret ingredient is missing: the user's password.

4. Information sharing with other users

With the aforementioned procedure, the process of sharing information with users who already have a Zivver account is relatively easy from a cryptographic perspective. It is done according to the steps described in the following sections.

4.1 Cipher key generation

A new AES-CTR 128 bit cipher key is generated for every file or message.

4.2 Symmetric encryption of file and message

The file or message is symmetrically encrypted with the generated key.

4.3 Asymmetric encryption of cipher key

TThe generated key is (asymmetrically) encrypted with the public key of all users with whom information is shared with.

We do not encrypt the files directly with the public keys of recipients (and sender), because asymmetric encryption is a computational heavy operation. As users might share large files with Zivver (up to 1 terabyte), asymmetric encryption would be both resource and time consuming. Symmetric encryption (using AES-CTR) is computationally much more efficient. When combined with the asymmetric encryption of the symmetric key, a high level of security is maintained.

For symmetric encryption we use 128-bit keys, also for computational performance. Although AES also has 192 and 256 bit options, experts agree that the level of security gained by the increased bit size is purely theoretical until quantum computing arrives, while significantly computationally more heavy (see here and here). Once quantum computing really sees daylight, we will upgrade to either AES-256 or other encryption algorithms to be released.

5. Information retrieval by users

For users who already have a Zivver account, information retrieval is performed according to the following steps:

Login and derived key retrieval

Every user has to log in according to the procedure described in section 2. User Authentication. In a login call, the API-client, on behalf of the user, should provide the user's password as one of the API-call parameters. Upon a successful login call, e.g. the Zivver platform takes the provided password and uses that as an input parameter to recreate the derived key, generated with PBKDF2 described in step 4.4. Once the derived key is recreated, it is returned as a result parameter in the original login call.

Information retrieval

The API-client is required to provide the derived key, obtained in step 6.1, as a part of every future API-call that is associated with performing authenticated actions. Upon retrieving a message or file, Zivver takes the derived key from the authentication bearer and uses it to decrypt the private key of the user according to the steps described in section 4.5. Subsequently it retrieves the symmetrically encrypted message or file from the object store, and uses the user's decrypted private key to decrypt the key of the message or file.

In the instance of a message, the decrypted body is returned as a response to the API-call. In the instance of a file, the API returns a temporary signed url allowing the client (browser) to download the file within a short period of time.

All of the above is performed fully in-memory. This means that the password, derived key, private key nor the message or file are never stored and are thus never available 'at rest'. This way, Zivver ensures that information is only readable by the intended recipients, and not by us or anyone else.

6. Information sharing with guests

Information sharing with guests is far more challenging. Guests do not have an account and thus do not have a public/private key pair to encrypt and decrypt their information. In order to securely share information with guest users and to ensure that Zivver do not hold decryption keys, we apply the following approach:

6.1 Provide guest information and access rights

When sending information to a guest, Zivver users specify a) an email address (and optional name) and b) the access rights the guest user must meet in order to read the message. Zivver offers various access rights for guest users:

- SMS-code: The sender can specify a mobile phone number to which Zivver sends a TOTP based access code to when the guest user attempts to access the message. For security reasons, Zivver does not support VOIP-phones at this time.
- Access code: Zivver allows (members of) organizations to specify an (organization wide), guest specific, access code, to be (re)used by users who belong to that organization.
- Email verification: In case the sender does not have access to a mobile phone, does not have an (organization) access code, and the user does not know how to communicate a (personal) access code to the recipient, Zivver enables users to utilize email verification. With this option, the recipient receives a notification mail which includes a link. Upon clicking the link, the recipient receives a second email with a temporary access link. Using this option reduces the interception risk of a normal email significantly, but is not equally secure as second factor authentication; access to someone's email inbox is sufficient to read a message.

6.2 Create new public/ private key pair per guest/ conversation

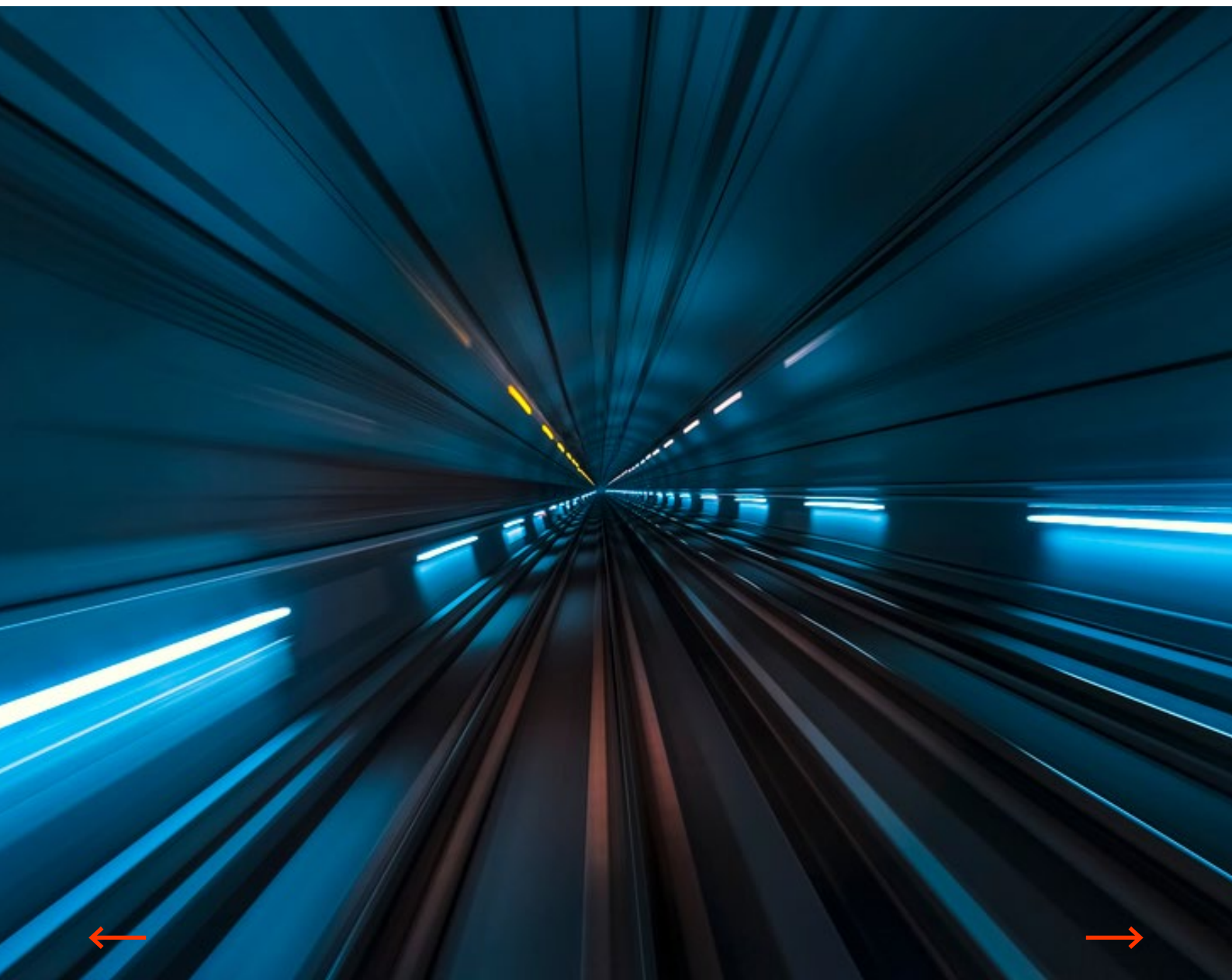
For every new guest added to a (new or existing) conversation, a new public/private key pair is created for use of the RSA-algorithm (Rivest–Shamir–Adleman, 2048 bits).

6.3 Store public key for message and file encryption

The public key of the user is stored for later use to encrypt files and messages in that same conversation.

6.4 Generate symmetric key and encrypt and store guest-conversation specific private key

Zivver subsequently generates a new AES-CTR 128 bit cipher key with which to encrypt the guest-conversation specific private key. The encrypted private key is subsequently stored. The cipher key is not stored, but used in the next step.



6.5 Send out notification email with symmetric key and username

For guests, Zivver sends a (styled and possibly organization branded) notification mail, informing them that a specific user has sent a secured message via Zivver. This email contains a link to the specific message/conversation.Tghe URL-parameters in the link contains the cipher key with which the guest-conversation specific private key can be decrypted. In addition, the notification email includes information regarding access rights selected by the sender (see section 7.1).

6.6 Wrap private key with public keys Zivver users for replies

When a user needs to reply in a conversation in which a guest participates, it is not possible to send the guest another notification mail about the new message as this would include the guest's private key for that conversation. However, the key was not stored to prohibit us as a service provider from having possible access to the information. To manage this issue, the symmetric key (from section 7.4) of the conversation for a guest is wrapped (encrypted) with the public key of every user and guest in that conversation. That way every one that could possibly add a reply to that conversation is able to cryptographically access the private key of the concerning guest, which we can subsequently include in the notification email as described in section 7.5.

7. Information retrieval by guests

Information retrieval by guests (users that do not have a Zivver account, but received a notification mail from Zivver) works according to the steps described in the following sections.

7.1 Guest clicks on notification mail

As described in section 6.5, guest users receive a notification mail informing them about a new message sent to them via Zivver. This notification mail includes a link in which the symmetric key is embedded. With this key the guest-conversation specific private key can be decrypted and the email address the mail was sent to. Clicking the link will redirect to the Zivver webapplication where it will extract the symmetric key and email address from the link.

7.2 Guest is logged in to Zivver

With the symmetric key and email address from the link (see above), the Zivver web application makes a login-call to Zivver. If the sender has set up an access right for the recipient (see section 7.1), the guest is subsequently challenged to proof his/her 'possession' of the specified access right.

7.3 Guest can access information

If successful, the user is logged into Zivver and has, via the Zivver web application, access to the message and any attached files by providing the symmetric key to the Zivver platform with every API-call. Therefore, decryption of the private key and subsequently the message and/or files can be performed in memory and served to the user via the client.

8. Master key options for organizations

The approach described above ensures that no-one but the user has access to their private key. This, however, is not ideal for organizations.

In order to fully take this responsibility, organizations must be able to monitor and even access the information their employees share. With other popular encryption services, this is generally not possible. Their encryption is done on client-devices with public keys, or recipients/participants and data is not centrally stored. Therefore organizations are unable to monitor or access the information their employees share. Zivver overcomes this limitation by using a master key construction.

8.1 Creating a new organization with an admin

Any user can create a new organization. When this happens, the user is given administrator rights for that organization.

8.2 Create new public/private key pair

For every new organization, a new public/private key pair is created. Encryption is achieved using the RSA-algorithm (Rivest–Shamir–Adleman, 2048 bits).

8.3 Wrapping the private master key with the admin key

The private master key of the organization is wrapped (encrypted) with the (public) key of the creating admin.

8.4 Store public key of organization

The public key of the organization is stored. This key is used to potentially add users to an organization (and wrap their private keys, as below). With the steps above, the following operations are possible for admins:

8.5 Add users to organization and wrap private key

Admins can invite users to join their organization. Admins can also create new user accounts and automatically add them to their organization if they belong to their organization (e.g. have an email address with a domain that corresponds to a domain the organization has claimed1). If a user accepts the invite, or when a new user is created within the organization, the symmetric key with which their private key can be decrypted is wrapped (encrypted) with the organization's master key.

8.6 Assign admin rights to other colleagues

Admins can assign admin rights to other users within their organization. In this case, the organization master key is decrypted with the private key of the current admin, and the organization's private key is subsequently wrapped with the (public) key of the newly assigned admin.

8.7 Retract admin rights from other colleagues

Admins can demote admins to normal users. In that case the user in question (public) key wrapped organization master key is deleted.

With the master key principles, Zivver cryptographically enables admins (those users that have a (public) key that is a wrapped version of the organization master key) to gain access to the sent or received information of users. With these possibilities admins can:

8.8 Regrant access to the message history of a user

As described in section 4, the password of the user is the key to access their private key, which is required for message and file decryption. In case a business user loses their password (e.g. when organizations do not use SSO, or SSO-configuration is lost), the user can not regain access to the message history. As Zivver does not have access to the user's private key, we cannot grant user access. However, as the admin indirectly has access to the user's private key, they are able to do so. Zivver thus provides admins with the option to:

i decrypt the derived symmetric key from the old password using the master key

ii decrypt the derived symmetric key from the new password

iii decrypt all old private keys with the old derived key

iv re-encrypt all these private keys with the derived symmetric key from the new password

Users can then re-access their messages and files sent and received before their password was reset.

8.9 Gain access to other users' sent and received information

In the instance of a data breach or fraudulent action of a user, Zivver provides admins with the possibility to gain access to a user's messages or files by:

i allowing admins to add themselves as delegates to the user's account

ii login in to the user's account via account delegation

iii viewing the messages and/or files with the delegated access

All of the above actions are logged by Zivver into an organization specific audit and communication log, available to all admins of the organization.

1. Zivver allows admins to claim the domains (e.g. top-level domains). This is done by sending a verification code to one of the following email aliases: admin@, administrator@, hostmaster@, postmaster@ or webmaster@the domain to be claimed. That verification code can be entered in the Zivver web application (or by any other API-client) which Zivver considers proof of the possession of the specified domain.

zivver

Zivver
59-60 Gainsborough House,
Thames Street Windsor,
Berkshire, SL4 1TX
United Kingdom

+44 (0) 203 285 6300
contact@zivver.com

www.zivver.com