zivver

Zivver's security and privacy by design

approach explained

Whitepaper

Table of contents

Introduction

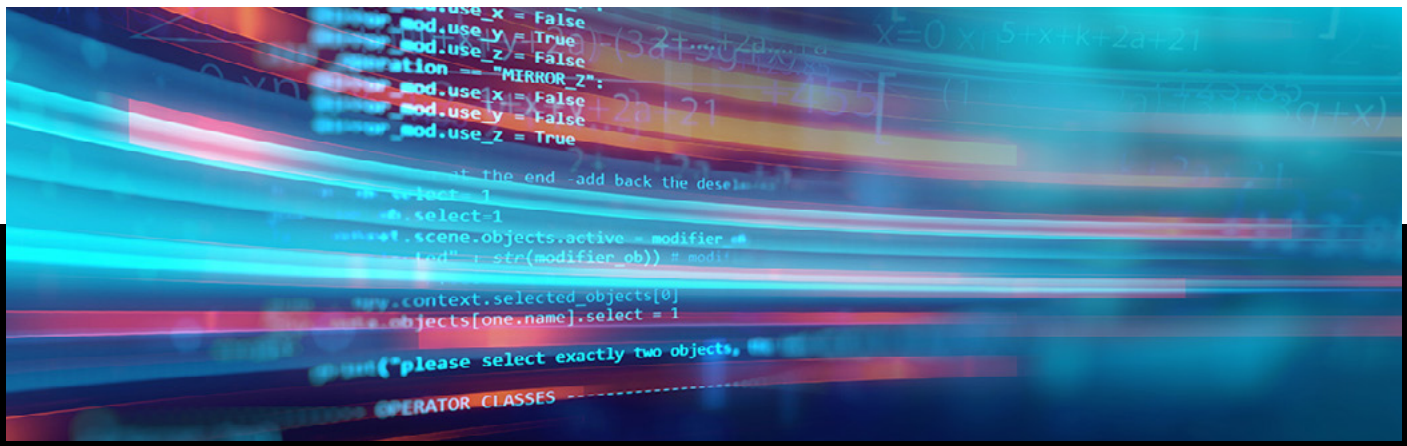
Most people think of encryption when talking about secure communication. They think of hackers being their primary risk. However, in practice, most data leaks are caused by human errors, like sending information to the wrong person.

So looking at encryption alone is a very limited way of looking at secure communication. It only limits the 'during' sending risk, while keeping the human errors 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 for most people (understandably) limited. In this white paper we outline how Zivver uses encryption to ensure that only the (intended) sender and recipients can read the secure messages sent via our platform.

A quick Google search would define 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." And exactly that is what we do. On a daily basis we do our best to ensure that the data of our users and also their contacts are safe. That means that we believe that sensitive data should also be unavailable to Zivver. This to ensure that we become neither a possible source of data leaks, nor an interesting target for hackers.

Making sure that we cannot access sensitive data ourselves requires that we don't have possession of the keys needed to decrypt information. This requires the use of encryption algorithms that rely on separate keys for encryption (public key) and decryption (private key), also known as asymmetric encryption. However, strong encryption requires the use of strong keys, in other words a very long password.



We aim to provide both users and organizations with tools that finally make secure communication simple.

For every company that takes security seriously, this poses a challenge: the provider cannot store the key, but it is needed when the user needs to decrypt the data. What is a feasible way to handle key management, while remaining both user friendly and secure? This is something 'traditional' encryption methods like 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 and creative solution.

In this white paper we outline the principles and functionality of this solution in all its facets. We'll touch on the following topics:

01 Transport Layer Security

02 User authentication and log-in

03 Public/private key creation and management for new users

04 Information sharing with other users

05 Information retrieval by users

06 Information sharing with guests

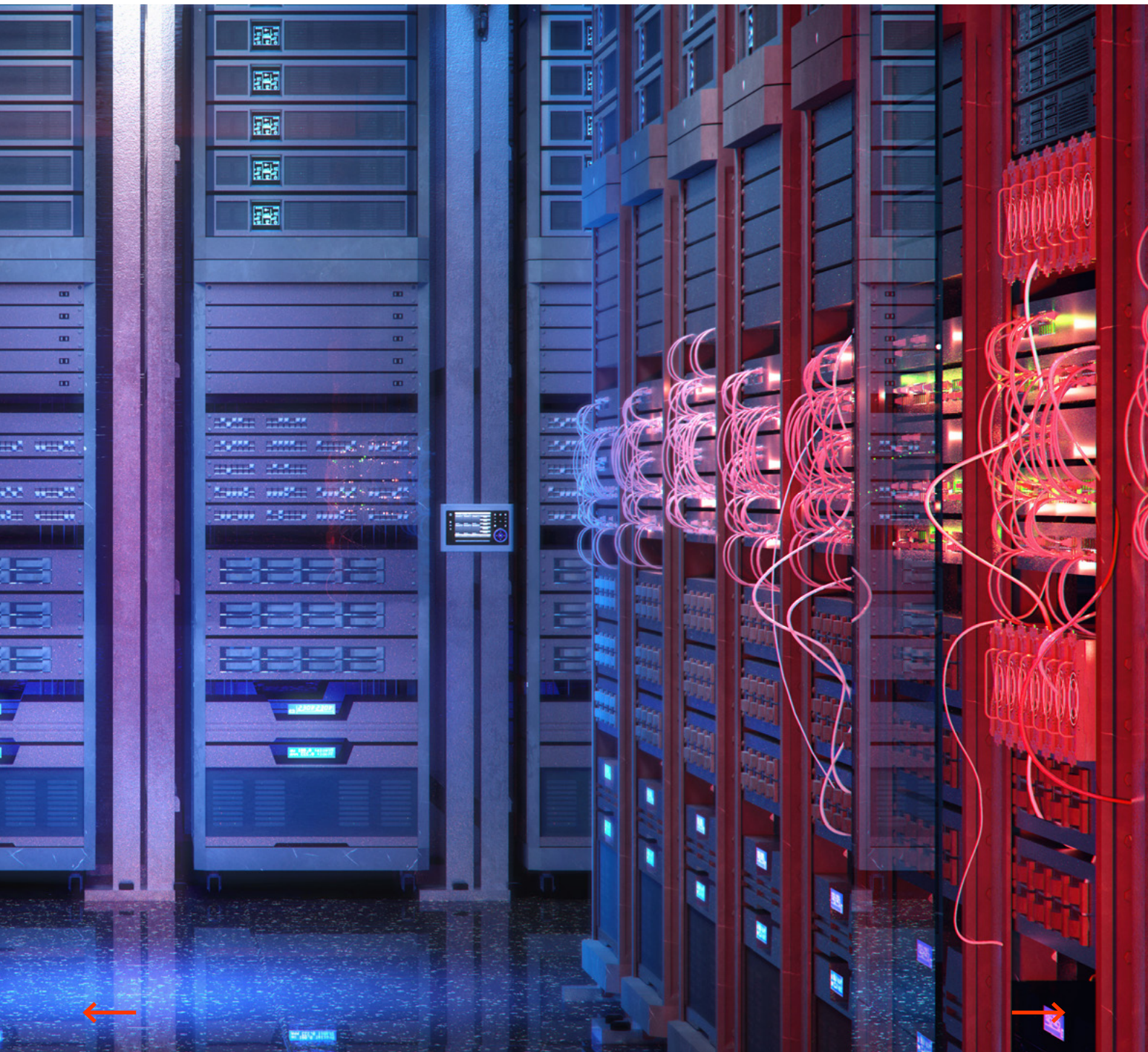
07 Information retrieval by guests

08 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.

01 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's the most widely deployed security protocol today. It can be used by web browsers and other applications that require data to be securely exchanged over a network.



02 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. In 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 via e.g. 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 usually 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.

Would you like to know more about hashing? Read our blog about it here.

2.2 Second factor authentication

Zivver only allows its 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 that is compatible with the TOTP-standard (RFC6238). Users are asked 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 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 minutes. If the access token is expired, the refresh token must be used to obtain a renewed access token.

03 Public/private key creation and management for new users

The basis of the security of the Zivver platform is formed by the use of a public-key cryptosystem in combination with some other methodologies to comply as much as possible with the zero-knowledge aim of Zivver. 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 done 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),

- Password is the user's password from which a derived key is generated,
- Salt is a sequence of bits, known as a cryptographic salt,
- c is the number of iterations desired, 65,536 in our case,
- dkLen is the desired length of the derived key, in our case 256 bits.

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, 256 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 needed, including the user's password and user's private key from memory.

The steps above are conducted fully in-memory in (any of the) Zivver servers. Therefore neither the user's password nor the user's private key is ever stored. It is never accessible for anyone while 'at rest'. As an analogy for non-technical users: together with the user we have prepared a nice meal and have written down the full recipe. However, we cannot prepare the recipe again without the user's presence, as the secret ingredient is missing: the user's password... How about that?

04 Information sharing with other users

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

4.1 Cipher key generation

A new AES-CTR 256 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

The generated key is (asymmetrically) encrypted with the public key of all users the 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 (> 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 256-bit keys. Historically Zivver has used 128-bit keys up until 2023. The decision to change this was based on recent recommendations from the NIST, and the dutch government.

05 Information retrieval by users

For users that already have a Zivver account, information retrieval is done according to the steps described in the following sections.

Login and derived key retrieval

When coming to the platform 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 of course, 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's 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 a file, the Zivver platform takes the derived key from the authentication bearer, and uses that 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 case of a message, the decrypted body is returned as a response to the API-call. In case 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. So neither the password, derived key, private key nor the message or file were ever stored and are thus never available 'at rest'. This way we can ensure that information is only readable by recipients who the sender has given explicit access to the information, and not by us, nor anyone else...

06 Information sharing with guests

Information sharing with guests is much more challenging: guests do not have an account and thus do not have a public/private key pair to encrypt and decrypt their information with. To be able to securely share information with guest users, while respecting the fact that we as a service provider should not be in possession of the decryption keys, we apply the following approach:

6.1 Provide guest information and access rights

When sending information to a guest, Zivver users should specify a) his/her email address (and optionally name) and b) the access rights the guest user needs to challenge before being able to read the message. Zivver offers various possible access rights for guest users:

- **SMS-code**: The user that sends the message can specify a mobile phone number to which Zivver sends a TOTP-based access code upon the attempt of the guest user 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 that belong to that organization. Think of using a patient- or client number as an organization access code in case of a guest of whom the phone number is unknown.
- **Email verification**: In case the sender has no phone number, no (organization) access code and the user does not know how to communicate an (personal) access code to the recipient, Zivver allows users to fall back on using email verification. With this option, the recipient will receive a notification mail with a link, and after clicking the link, he/she receives another mail with a temporary access link. Using this option will reduce the interception risk of a normal e-mail significantly, but is not equally secure as second factor authentication; access to someone's e-mailbox is sufficient to read a message.

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

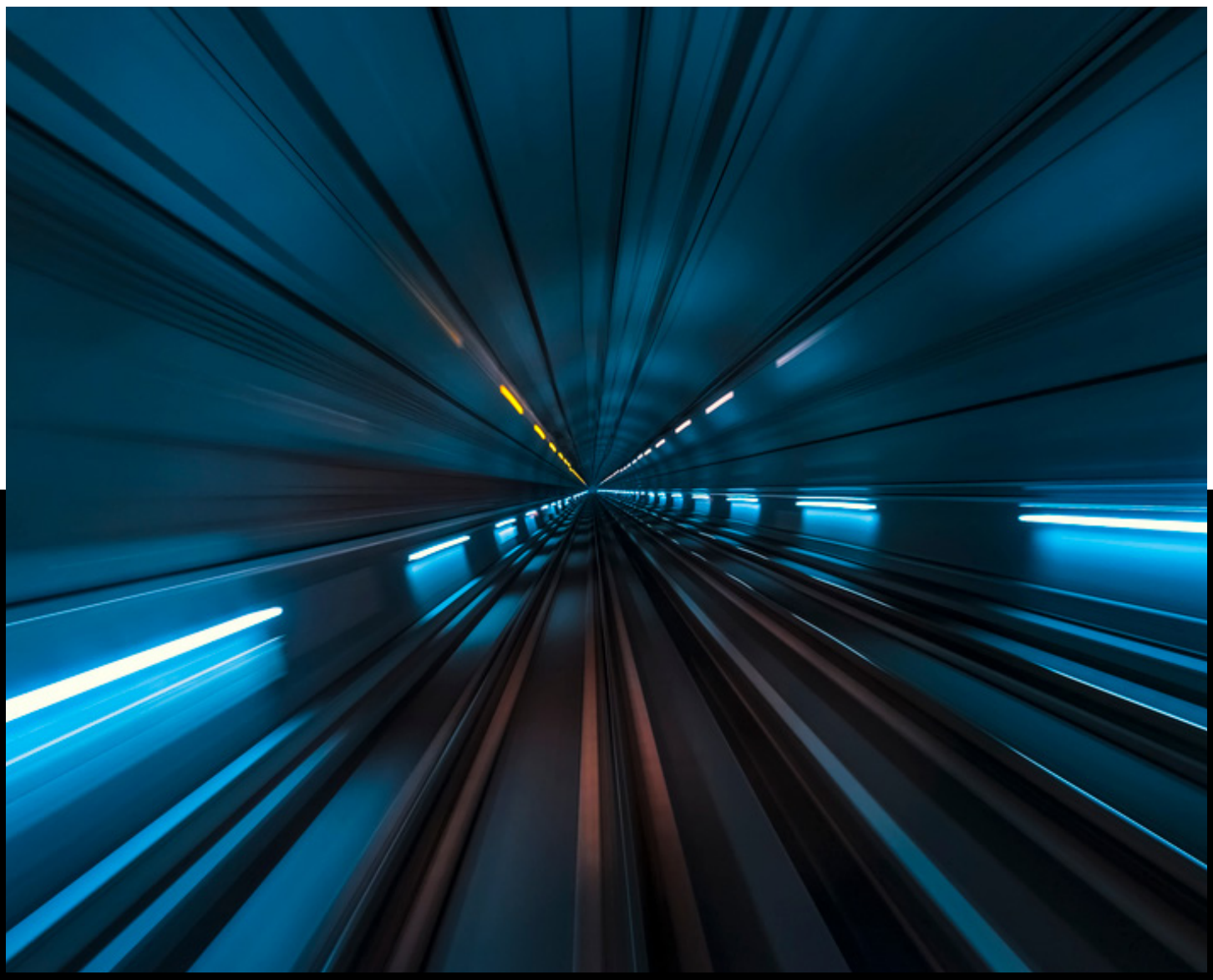
For every new guest that is added to a (new or existing) conversation, a new public/private key pair is created for use of the RSA-algorithm (Rivest–Shamir–Adleman, 2048 bits).

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

We subsequently generate a new AES-CTR 256 bit cipher key with which we 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, we send out a (styled and possibly organization branded) notification mail, informing them about the fact that a specific user has sent him/her a secured message via Zivver. This email contains a link to the specific message/conversation. On of the URL-parameters in the link contains the cipher key with which the guest-conversation specific private key can be decrypted. In addition, the notification mail includes information about access rights that the sender selected (see section 7.1).

6.6 Wrap private key with public keys Zivver users for replies

Until here, the process of sending a message to guests wasn't that complicated (conceptually). However, how to handle the situation that a user wants to reply in a conversation in which a guest participates? In that case you would want to send the guest another notification mail about the new message. However, that would require also including the guest's private key for that conversation. But that key was not stored to prohibit us as a service provider from having possible access to the information. To deal with this issue we implemented the following solution: 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, can cryptographically access the private key of the concerning guest, which we can subsequently include in the notification email as described in section 7.5.

07 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 webapplication 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 webapplication, access to the message and (possibly) attached files by providing the symmetric key to the Zivver platform with every API-call. So 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.

08 Master key options for organizations

The approach described above ensures that no-one but the user has possession of (access to) their private key. This is however not a desired situation for organizations.

From a legal perspective the organization a user works for is responsible for the actions of the users and the possible data leaks they cause.

To be able to fully take this responsibility, organizations should have a possibility to monitor and even access the information their employees share. With other popular encryption services this is generally not possible. Their encryption is done at the client-devices with public keys or recipients/participants and data is not centrally stored. Therefore organizations can never monitor, let alone access, the information employees share. Zivver overcomes this limitation by using a master key construction that is described in the following sections.

8.1 Creating a new organization with an admin

Any user can create a new organization. When this happens that 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 done 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 possibly add users to an organization (and wrap their private keys, see step 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 auto-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 that 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 concerning user's (public) key wrapped organization master key is deleted.

With the master key principles, Zivver cryptographically enables admins (better, 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 his/her private key, which is needed for message and file decryption. In case a business user has lost his 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 we do not have access to the user's private key, we cannot give users the access back ourselves. However, as the admins indirectly have access to the user's private key, they can. Zivver thus provides admins this option, which basically is:

- decrypt the derived symmetric key from the old password using the master key.
- decrypt the derived symmetric key from the new password.
- decrypt all old private keys with the old derived key.
- 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 case of a suspicion of a (serious) 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:

- allowing admins to add themselves as delegates to the user's account,
- login in to the user's account via account delegation,
- viewing the messages and/or files with the delegated access.

All of the actions above are logged by Zivver into an organization specific audit and communication log, that is available to all admins of that 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 webapplication (or by any other API-client) which Zivver considers proof of the possession of the specified domain.

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

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

www.zivver.com