Trustner

Simple and Secure Communication

Whitepaper on Security

Version 1.3.

Contents

1 Background

Trustner is a simple and secure communication service.

It is understood that there is no absolute security on the Internet. Trustner can, for instance, only be as secure as the user's computer. Other risks must be taken into consideration as well.

Still, Trustner is a major security improvement in electronic communication as it routinely encrypts data in a way that is considered state of the art. When the user is careful with his or her computer (i.e. regularly updating antivirus protection) the overall risk of privacy infringement is reduced considerably. Even if the infrastructure of Trustner is taken over by an outside force, the attacker cannot easily gain access to content since the private keys are stored only on the user´s devices.

Trustner is a German company and based in Germany, where legislation on data protection is well developed. The location is seen as a competitive advantage.

It is important to understand that Trustner is **not** for anonymous communication. It is for electronic exchange among and with professionals in healthcare, the legal sector, banking and other sectors where confidentiality is required. Some Trustner functionality requires audit-proof authentication of users.

In privacy-sensitive areas like healthcare, German law requires a so-called Data Protection Concept (DPC) to be presented to the privacy authorities before software can be put into use. In cooperation with careon, an eHealth specialist, Trustner has prepared a DPC for Trustner Medical Apps to be used by doctors who want to communicate with colleagues, patients, their family, and other medical staff.

The Trustner DCP has been presented to data protection authorities of several *Bundesländer* (federal states), where Trustner has been piloted.

This document mainly describes the encryption concept of Trustner, which is an important part of the DPC. The encryption concept itself has been developed in cooperation with a senior encryption expert from the Horst Görtz Institut für IT-Sicherheit at the Ruhr-Universität Bochum, one of the leading German research institutions for data security.

For readers who want to know more than is described in this document may contact Trustner.

2. The Trustner Platform

2.1 Some Definitions

2.2.1 API

The Trustner API provides a RESTful interface for the Trustner platform. The platform stores encrypted content and the relationships between content and user, i.e. the link between a post and a room or a chat.

2.1.2 Apps

In Trustner the user can choose from a selection of different communication apps (or “room types”). Some apps allow data uploads; other apps delete content automatically after some time. There are apps for doctors which can only be used with personal identification. Apps run on a web server and use Trustner platform services. Trustner apps may be hosted and operated by a competent third party.

2.1.3 Client

There is a Trustner client for various devices and operating systems (Windows, OSX, iOS, Android). The encryption component of the client is called KeyBox.

2.1.4 Core

The Trustner core constitutes the basic user-interface outside the app interface. Responsibility lies with Trustner.

2.1.5 Instance, “Room” or Chat

A Trustner app can have an unlimited number of instances that is “rooms”. A room is a virtual meeting place where users communicate or share data. Every room can have an account- specific user with its own user profile. A chat is a basic room for direct communication between selected contacts.

2.1.6 Store

The Trustner Store is the place where the different apps can be activated. Some apps are for free while others have to be paid.

2.1.7 Platform

The Trustner platform manages user authentication, invitation mechanisms, app administration, and storage and payment methods as services. It also stores and distributes the public keys of the registered users.

2.2 The Concept

Trustner is built as a platform. It enables developers to create communication apps for users with different privacy and contact requirements, to be run independently from each other.

All apps, from internal as well as external developers, use the same API. However, the initial version has an API that is open only to internal programmers. As the platform progresses, third party developer accounts will become available.

All apps, regardless of who created them are accessible to users in the Trustner Store, which is part of the core.

The main function of the platform is, in conjunction with the Trustner client, the provision of encryption functionality.

The secret is encrypted and stored on the user device´s hard disc only. The secret for encryption and decryption of the locally stored data is provided by the Trustner core after user authentication.

Each device has an asymmetric key pair consisting of a private and a public key.

The so called resource key is a symmetric key for encryption and decryption of data within the Trustner app. This key is known by all members of an app instance. There is no limit to the number of resource keys in an app.

The Trustner client automatically manages the exchange of the symmetric key among the contacts in a room or a chat.

3. The Trustner Client

The encryption of Trustner is client-based and uses standard open source libraries.

The Trustner client organizes the encryption and decryption of shared content. It also handles the exchange of cryptographic keys between the Trustner apps. The resource keys are encrypted and stored in the client’s local database.

The client looks and feels like a web browser; it uses web technologies. It has exclusive access to Trustner and the Trustner apps. External URLs are handled by the user's web browser.

3.1 Cryptography

Communication between Trustner and Trustner apps and communication between Trustner apps and the end device, as well as vice versa, is encrypted using TLS 1.2.

There is no master key for key recovery. An optional procedure to provide a second key through a fully automated process with no access for the provider can be implemented at a later stage.

The symmetric encryption uses AES 256 (OFB mode) with OpenSSL Version 1.0.1f.

For asymmetric encryption of the resource key RSA with a key length of 2048 Bit is used. Elliptic curves as an alternative to RSA are being considered.

Encryption algorithms are versionized to allow an easy replacement if necessary.

3.2 Operating Mode

The download of the Trustner client (its encryption component is called KeyBox) initiates a registration procedure. Repeat login after successful authentication opens a user-specific client session. A session-token is generated on the server, which is then relayed to the KeyBox by loading the site using a JavaScript-invocation.

The session token establishes an association between the user account and the KeyBox.

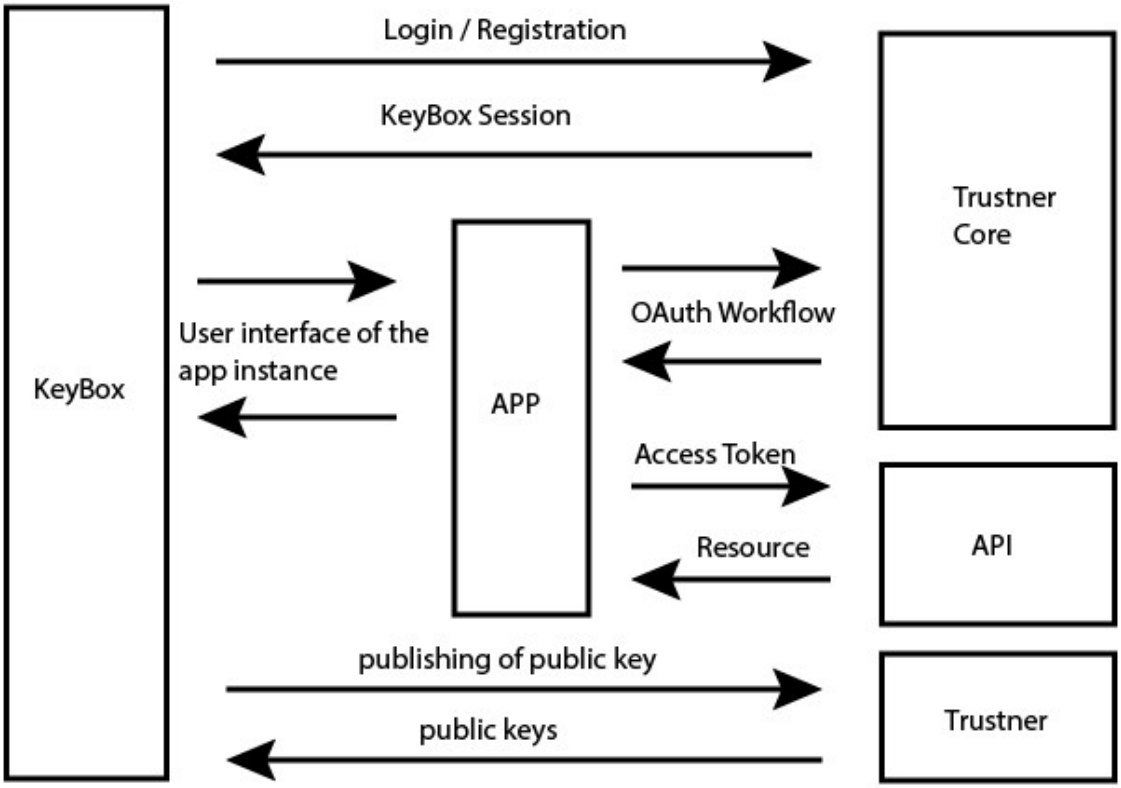

Figure 3.1: Exchange between App, Core, API, KeyBox and Trustner

Figure 3.1 shows the exchange including the OAuth workflow for third party apps, which has not been implemented.

3.2.1 Case 1: A public/private key pair does not yet exist

On the user's device, a public/private key pair is generated in the KeyBox. The default setting is a RSA 2048 key pair.

The time required for the generation of the keys depends on the entropy-pool of the user's computer.

After successful generation, the public key is rolled out, the private key remains in the client encrypted with the secret.

An indicator at the lower margin of the client-window notifies the user, when encryption is available.

3.2.2 Case 2: There is already a public/private key pair for the user

The private key is encrypted using a secret and hence secured. The session token enables the KeyBox to retrieve this secret through the Trustner API. The private key remains exclusively on the user's device.

There are two options envisaged for the generation and storage of secrets:

- In normal mode, the secret will be generated by the client and stored only on the server. This protects the private key against offline-attacks. Since the secret is retrievable through the users password a sufficient level of password quality is enforced in the account registration process.
- In advanced mode, the user is able to individually choose the secret, which is then only located with the user. In addition to the log-in password, a second password exists that serves for the decryption of the private key.

Only with a decrypted private key the resource keys can be decrypted. And thus the contents of the apps can also be decrypted. Since the resource key is a symmetric key it will be used for encrypting the contents of the apps. The resource key is encrypted with the public keys of all recipients.

In case a KeyBox is missing a resource key it can issue a key exchange request to the other KeyBoxes associated with the app instance.

3.3 Trustner Client API

The KeyBox API is accessible for apps via a JavaScript bridge. It is accessible only through a Trustner client view. It can only be used internally in Trustner apps and in the Trustner core. The API includes methods for encryption, decryption and key exchange.

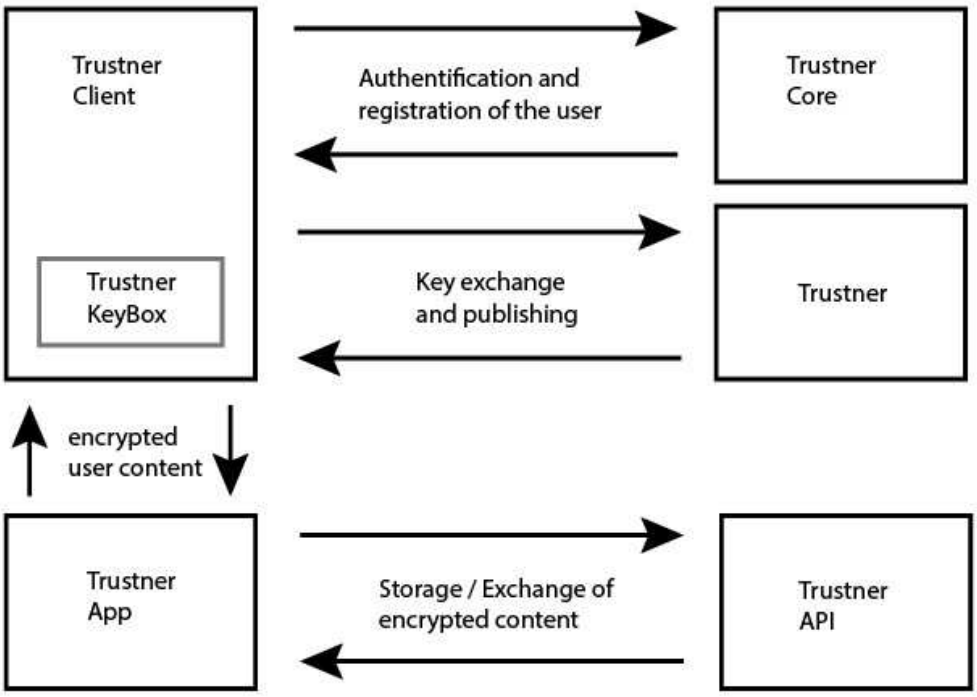

Figure 3.2: Operating Mode of Trustner KeyBox

3.4 Trustner Client Authentication Protocol

The Trustner KeyBox is associated with a user account after either the login or registration process. The KeyBox is connected with a user account through a unique identifier: the KeyBox ID. A user can have several KeyBoxes and for each KeyBox ID there exists a public-private-key pair.

A KeyBox installation can be shared by many users. However, using the Trustner KeyBox on a public computer, for instance in an internet cafe, is discouraged.

The private key is protected by the secret as defined above.

After login the Trustner KeyBox is issued a Trustner API access token.

3.5 Key Exchange

In order to be able to communicate in an app instance with two or more users, symmetric resource keys have to be available to all members of an app instance. It is ensured that these resource keys are not intercepted by either a third party or a Trustner administrator.

This is achieved by encrypting the resource key with the public keys of all members of an app instance. The KeyBox on the user´s device performs the encryption.

Therefore, for encrypting the symmetric resource key it is crucial that the KeyBox knows the public key of every app instance user.

After generation, the public key of each KeyBox is transferred to the Trustner server.

Public keys can be asked for by other KeyBoxes. This is the case when a new member enters an app instance or when a new device is added to an account. The public key needs to be announced to them so that they can exchange resource keys.

The AES 256 key is stored as a 64-character long hexadecimal representation. The key is generated using the system's random number generator.

Apple, Android and Windows devices use the OpenSSL random number generator without the dysfunctional DUAL EC DRBG.

A newly generated resource key is encrypted for local storage using the public key of the user's KeyBox. For distribution the resource key is encrypted for each instance user with its public key. Permission to deposit a public key encrypted resource key is granted only to the members of the app instance.

The permission check is performed by the Trustner platform.

Every key exchange is logged in the user's client. Advanced users can check back on a log, in order to see who issued a key to whom. Later, advanced users may be able to individually permit a key exchange.

The decryption of deposited keys is only possible using the private key of the KeyBox.

3.6 Additional Clients

The user can use several devices to access Trustner apps. Each device requires a Trustner client download. With his credentials the user can access the apps from all his devices.

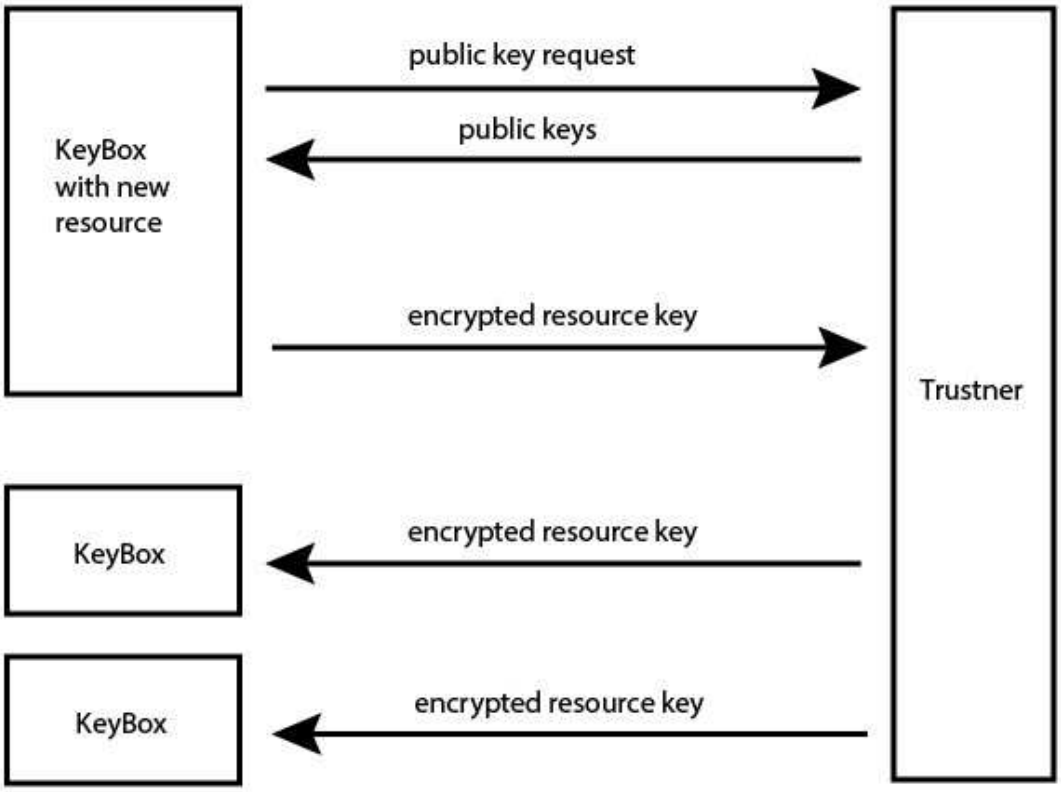

Figure 3.3: Exchange of Resource Key to new KeyBox

When it comes to key exchange, an additional device is treated like an additional contact in a room or chat. The user can see all his registered devices.

The protocol of every installed and currently used device is generated on the server and stored there.

4. Risk Assessment

Trustner offers automatic end-to-end encryption of communication content. It gives a very high level of protection against unwanted insights of third parties. However, the actual security level depends on several factors.

4.1 Weak Password

If a password can be guessed, the user will risk losing his privacy on Trustner.

Registration of a Trustner account requires a password that fulfills the requirements of the German *Bundesamt für Sicherheit in der Informationstechnik* (BSI): Passwords have to have a minimum of eight lower and upper case characters, and must include special characters and numbers – or be longer than 20 characters.

There is a time-out after three failed login attempts.

Passwords are stored with a one-way hash function (bcrypt) and are hardened with an additional individual Salt.

4.2 Password Theft

Even a strong password can be stolen when the device of the user is compromised.

The only defense is to create awareness at the user level. A user should never login to Trustner on a computer he or she does not trust.

The computer of the user needs to be regularly updated. An up-to-date antivirus protection program will reduce the risk of password theft. Regular password changes are strongly recommended.

4.3 Compromised Client-Server Communication

The risk of a man-in-the-middle attack is highest in Wi-Fi zones and public internet access points, where others can interfere without being noticed. In this case the attacker can compromise the client-server communication.

Trustner uses TLS 1.2 as a defense. However, given the situation with compromised certificates, a risk remains.

This risk is addressed through the option to check the fingerprint of the certificate.

4.4 Limited Randomness

Strong encryption relies on randomness in the process of key generation. If the random key space is somehow limited, i.e., through a manipulated entropy pool, a private key may be brute forced.

The OpenSSL random generator used by Trustner promises a sufficient degree of randomness.

4.5 Bad Trustner Client

There is no security when the user installs a Trustner client that has been tampered with. This can happen when the user downloads the client from a fake website.

It is therefore important that the user goes to the official Trustner website each time he logs on to Trustner.

Trustner publishes the fingerprints of the binaries via HTTPS. An advanced user may check those to check the integrity of the binary.

4.6 Compromised Client Device

Trustner has no control over the devices where the client is downloaded. While professional organizations may have staff to secure infrastructure and hardware, private users usually do not have that protection.

A compromised client device is hard to detect. The best way to avoid this is to raise user awareness for regular software updates. Also, a current antivirus program can help.

4.7 Compromised Trustner Infrastructure

An unnoticed attack on the Trustner infrastructure could result in the most serious damage. It would enable an attacker to target individual user accounts and their contacts and thus potentially all Trustner users.

The attacker would have to gain access to the resource keys by adding KeyBoxes to accounts. Since all key exchanges are logged on the client side, the user might eventually notice the addition of unauthorized KeyBoxes. However, a skillful attacker may find ways around this.

A safeguard for users is to limit the sign-in of new clients in a given app instance. That limit is stored on the client device. The option is offered as “close room or chat”.

5. Open Source Policy

The Trustner security concept is based on transparency and comprehensibility. Trustner uses only standard libraries and standard protocols. At a later date, the code of the key management process in the Trustner client will be published for public review.

6. Contact

Dr. Harald Sondhof

Trustner GmbH
Österbergstraße 9
72074 Tübingen
Germany

www.trustner.com