Technical white paper

# Encrypted messaging

October 2019

XING

# Contents

# Introduction

XING is the leading online business network in German-speaking countries. XING provides its members with instant messaging via mobile apps (Android, iOS, Windows Phone), B2B and B2C web clients as well as via API.

As of 26 March 2018, XING also provides B2C end-to-end encrypted messaging via its Android and iOS apps.

This document provides a technical explanation of how end-to-end encryption has been implemented in XING's messaging solution.

# Scope

XING's end-to end encrypted messaging pursues the following goals:

- Provide members with the option to send end-to-end encrypted messages
- Ensure that the existing messaging infrastructure is not changed in order to remain compatible with web clients and API consumers
- Build on established best practices for encryption

The first iteration of XING's end-to-end encrypted messaging involved the following parameters:

- Encryption should be available for one-to-one text messaging on Android and iOS devices. XING's web clients and API consumers are not supported
- Only one mobile device per user can be used for encrypted messaging. This device acts as the user's trusted device. Encrypted messaging can only be established between trusted devices
- Encrypted chat and message history is deleted when switching to a new trusted device. In such cases, secure sessions need to be re-established
- End-to-end encrypted chats need to co-exist alongside non-encrypted chats to ensure compatibility with existing consumers

The underlying protocol is designed with the following aspects in mind:

- Security – on account of the threat model (see below)
- Future extensibility in terms of multi-device or group-chat support
- Usability – keep friction to a minimum
- Reliability – foster and sustain user trust

## Trusted Device Setup

When the user logs in with the XING Android or iOS app for the first time, the app will register itself as the user‘s trusted device. In doing so, it generates a number of key pairs:

- A public “identity” key `(IKs, IKp)` for the user
- A public “device” key `(DKs, DKp)` for the device
- A batch of public “pre key” keys `(PKs, PKp)` for single use

All public keys are uploaded to the server, while all of the secret keys are stored in the app‘s protected internal storage. The combination of an identity key, a device key, an incremental revision number R, and the user‘s userid UID is known as the recipient’s “key bundle” and describes the recipient’s cryptographic endpoint.

Before being uploaded, the key bundle is signed by the identity key to guarantee its integrity when used by other clients, and to avoid scenarios like unknown key share attacks. Pre keys are intended for single use only, and the server delivers them just once for use during session setup. The list of pre keys on the server is continuously maintained by the client, replacing keys with new ones once they have been requested. Note that “device” keys are, at this point, only included for forward compatibility with a multi-device or group chat feature.

## Encrypted Chat Sessions

An encrypted chat is established between two devices and managed entirely by the client apps. It is encrypted between the endpoints in such a way that transferred data can never be read by other parties, including the relaying infrastructure.

In an encrypted chat, clients maintain shared cryptographic secrets in “sessions”. These sessions are uniquely identified and strictly ordered,

based on their creation timestamp. That way, there is always a newest session which will be used for encryption. Sessions are established unilaterally by the first party when sending a message, and expire when a user changes their trusted device.

## Session Setup

When a client sends its first message in an encrypted chat, or a previous session expires due to device takeover, it creates a new session.

The session setup process is a constellation of Diffie-Hellman key exchanges as follows:

1. The initiator requests the intended recipient's key bundle `(IKp, DKp, R, UID)`, a pre key `(PKp)`, and a timestamp `(T)` from the server. This key bundle is checked for integrity.
2. The initiator generates a new key pair `(EKs, EKp)` and loads its own identity key as IKs.
3. The initiator generates a session secret SK from a cryptographically secure pseudo-random number generator.
4. The initiator calculates a shared secret with the recipient:
   - `x = DH(IKs, DKp) || DH(EKs, IKp) || DH(EKs, DKp) || DH(EKs, PKp)`
   - `ESK = ENC(SK, x, T)` The timestamp `T` is included as additional authenticated data. This data is bundled into a "session setup block" `SSB = (EKp, PKp, T, ESK)`.
5. The session setup block is sent together with outgoing messages until a message using the same session is received.
6. All subsequent messages do not include the `SSB`, but only `T` to identify the session.

The pre key and ephemeral keys serve a similar purpose, which is twofold:

- First, the static keys make the exchange unique rather than deterministic
- Second, it mixes in transient information from both sides that can be deleted after the exchange has been processed, rendering the transmitted data completely useless afterwards

## Message Receipt

When a message is received with an unknown identifying timestamp `T`, the recipient sets up a new session from the `SSB` along with the message.

1. The recipient obtains `SSB = (EKp, PKp, T, ESK)` from the initiator
2. The recipient looks for the `PKs` corresponding to `PKp` in their local storage, and subsequently deletes the secret key material for this pre key from its database
3. The recipient loads its own keys `IKs` and `DKs`.
4. The receiver calculates a shared secret:
   - `x = DH(IKp, DKs) || DH(EKp, IKs) || DH(EKp, DKs) || DH(EKp, PKs)`
   - `SK = DEC(ESK, x, T)`

At this point, both the sender and recipient have a shared secret `SK`. This secret is stored together with `T` to identify the session.

## Participant Consistency and User Verification

Clients ensure participant consistency in the encrypted chat message history. To achieve this, they notify the user when sessions are invalidated, which occurs when their communication peer changes their trusted device. In such cases the key bundle revision will increase by 1, and a new session will be established. This property is strictly managed by the client, and does not rely on the server.

Users can mutually scan their QR code to verify participants. This QR code uses a 'xingauth' URI scheme. The data in the URI consists of the bytes of the hashed public identity keys of both participants in base64 representation. For hashing the individual public identity keys the Argon2i algorithm with 'INTERACTIVE' parameter defaults is used.

When a communication peer's code is scanned and successfully verified, the status of this secret chat session will be saved as verified. If the trusted device of such a peer changes, the user will be advised to renew their verification.

## Conclusion

XING offers optional encrypted messaging to users with a trusted device.

Encrypted XING messages between XING users are protected with an end-to-end encryption protocol so that third parties and XING cannot read them. This means that messages can only be decrypted by the recipient.

XING servers do not have access to XING users' private keys. XING users have the option to verify keys in order to ensure the integrity of their communication.

**New Work SE**
Dammtorstraße 30
20354 Hamburg
Germany

Phone +49 40 419 131-0
Fax +49 40 419 131-11
info@xing.com