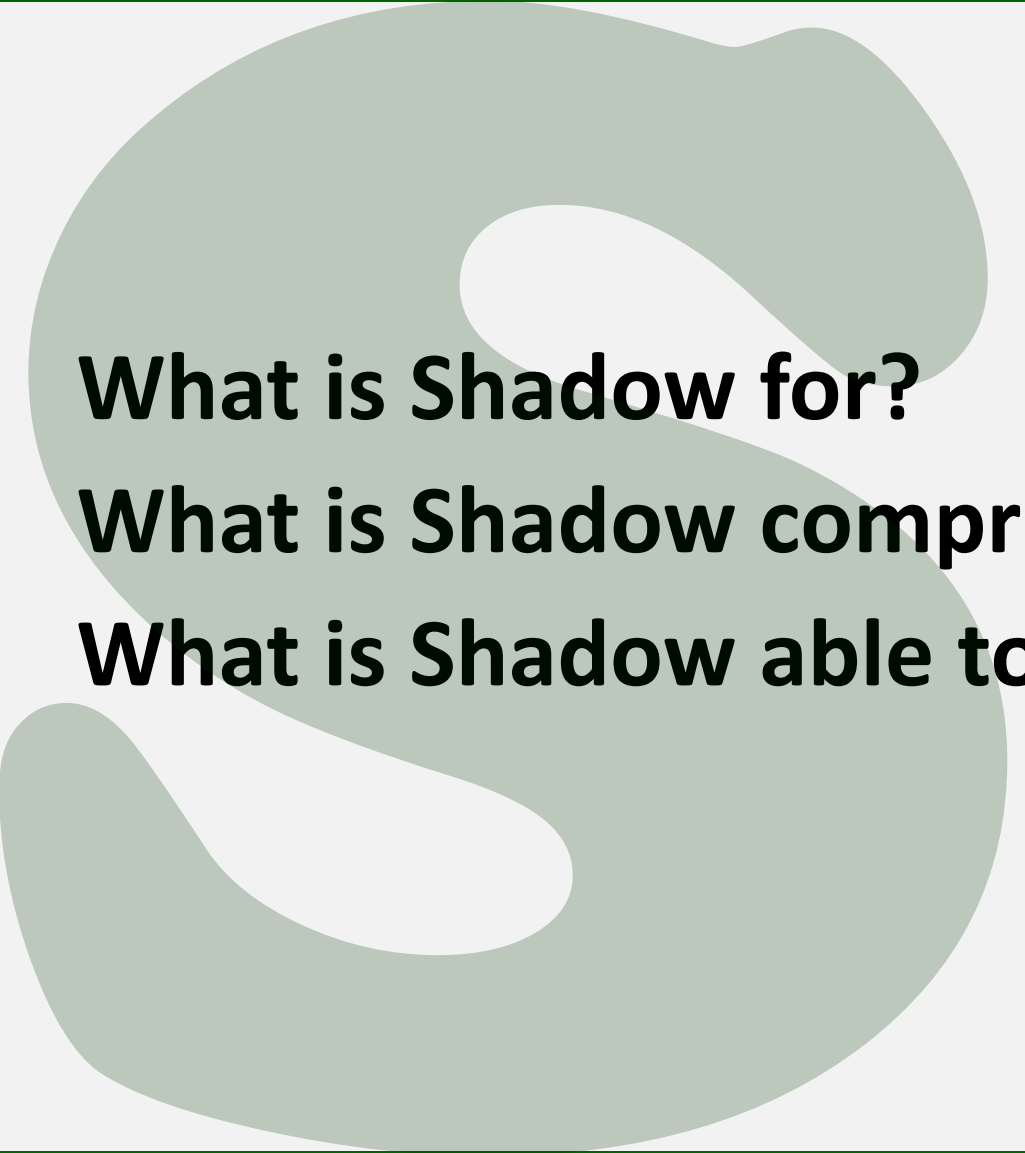




Secure Messenger
SHADOW

Open source. Full control. End-to-end encryption

- 
- 1. Mission:** What is Shadow for?
 - 2. Architecture:** What is Shadow comprised of?
 - 3. Functionality:** What is Shadow able to?

Why do we need a messenger?

- It is cheaper than phone communication
- It is more convenient than email
- It increases speed of communication and productivity
- It is more handy on mobile devices
- Nowadays messenger is a de-facto communication standard for mobile devices

Why do we need a secure messenger?

- Internet traffic in unprotected communication channels is comparatively easy to intercept for organized crime, unscrupulous competitors, raiders and foreign secret services
- Leakage of confidential messages, conversations, documents may cause significant material or reputation damage, become the instrument of blackmail - up to utter collapse of your business
- It is possible to secure communication by moving traffic to a VPN
- However, implementing a corporate VPN makes the system more complex and expensive in installation and maintenance
- It is simpler to implement the functionality of secure communication directly in the application code – that is, to incorporate it into the messenger itself

Why do we need end-to-end encryption?

- Encrypting traffic only between the client and the server means that encryption keys are known to the service provider
- Even if the server is your own, there remains the threat on the part of unscrupulous system administrators or hackers who gained access to the server
- The solution: end-to-end encryption. The keys are generated directly in messengers and are known only within those. One can only intercept traffic on the server, but not decrypt it.

Declaration of end-to-end encryption – is it a security guarantee?

- Can we blindly trust developers' declarations?
- Talking about “end-to-end encryption”, but what in reality?
- Threats of decryption on intermediate nodes, backdoors in code
- A vulnerability may stay open for years until the developer finds it out
- The solution: full openness of the source code, both for the client and the server
- Open source code means that:
 - independent analysis of utilized cryptographic protocols is possible
 - independent code audit to search for backdoors and vulnerabilities is possible
- Public access to the code means prompt detection of vulnerabilities by worldwide community of software developers and IT security professionals

End-to-end + Open Source – is it a security guarantee?

- What if the published source code does not match the code that is actually in service?
- One can build the client and verify checksums, but how would one verify the provider's server?
- What if the server's executable code is all right, but, at the same time, it grants opportunities to the service provider which are unwelcome by users (collecting registration data, communication session metadata...)?
- The solution: your own server with full control over it

An example: WhatsApp

- WhatsApp is a most popular messenger supporting end-to-end encryption. That said, its source code is not disclosed, and the servers are under administration of the provider itself
- *“German researchers say that a flaw in the app's group-chat feature undermines its end-to-end encryption promises”, - reports The Wired.*

“Anyone who controls WhatsApp's servers could effortlessly insert new people into an otherwise private group, even without the permission of the administrator who ostensibly controls access to that conversation. The confidentiality of the group is broken as soon as the uninvited member can obtain all the new messages and read them”.

(<https://www.wired.com/story/whatsapp-security-flaws-encryption-group-chats/>)

An example: Signal

- Signal is a messenger with end-to-end encryption and completely open source code
- In May 2018 cybersecurity experts detected a serious vulnerability in the desktop version allowing attackers to access other users' chat history by sending them a malicious code fragment with quote
- The experts notified the developers prior to publishing their report
- As a result of this co-operation, the vulnerability was promptly removed

Your own server as protection from user-side threats

- Centralized services with open registration suffer from threats from their own subscribers:
 - Spam
 - Distribution of illegal content
 - Exploiting vulnerabilities
 - Masquerading attempts
- Your own server with limited subscriber base of authorized users largely removes these threats

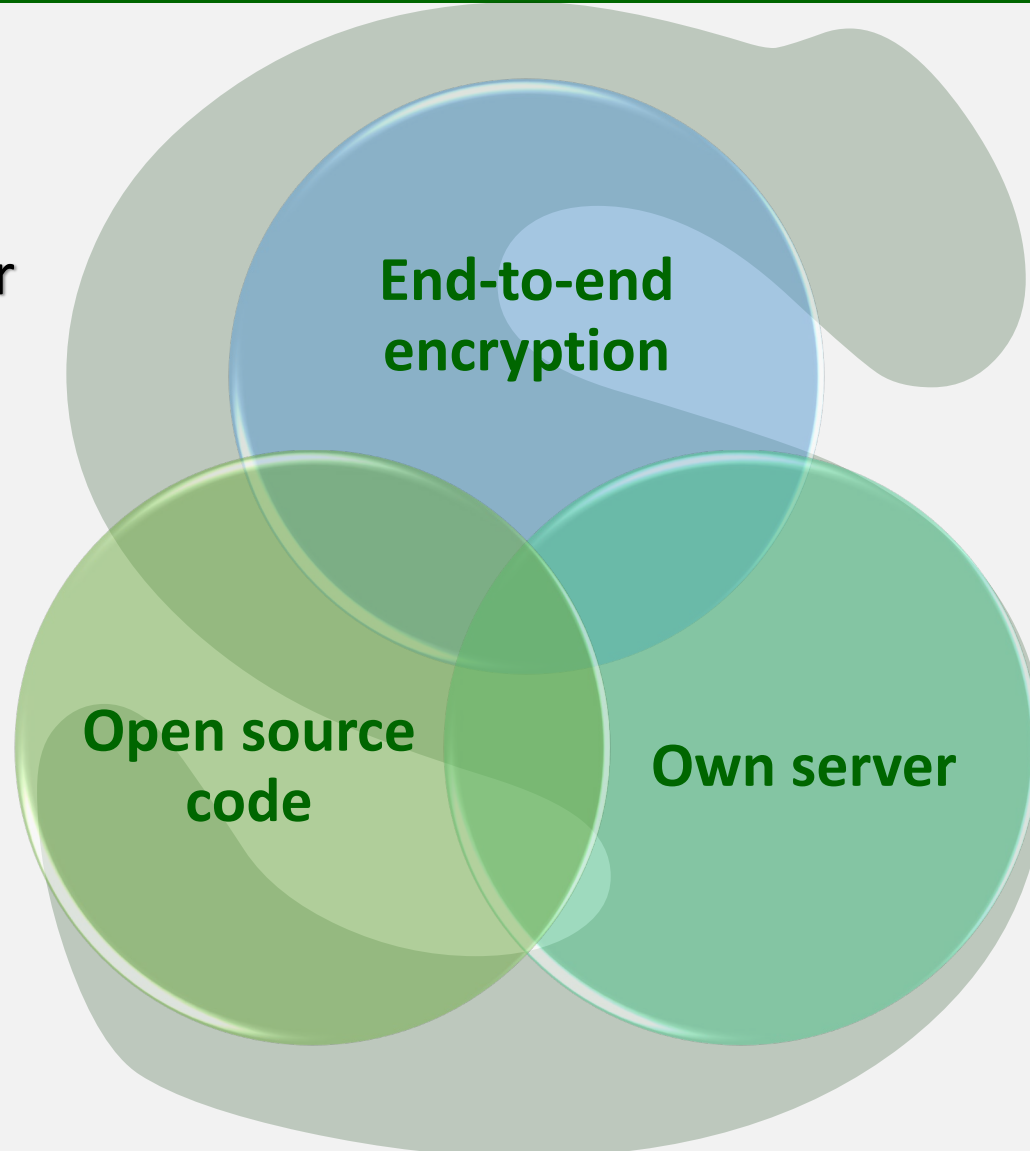
An example: WhatsApp

- *“A vulnerability in the messaging app WhatsApp has allowed attackers to inject commercial Israeli spyware on to phones.”*
- *“The malicious code, developed by the secretive Israeli company NSO Group, could be transmitted even if users did not answer their phones, and the calls often disappeared from call logs”.*

(<https://www.ft.com/content/4da1117e-756c-11e9-be7d-6d846537acab>)

Three key components of a secure solution

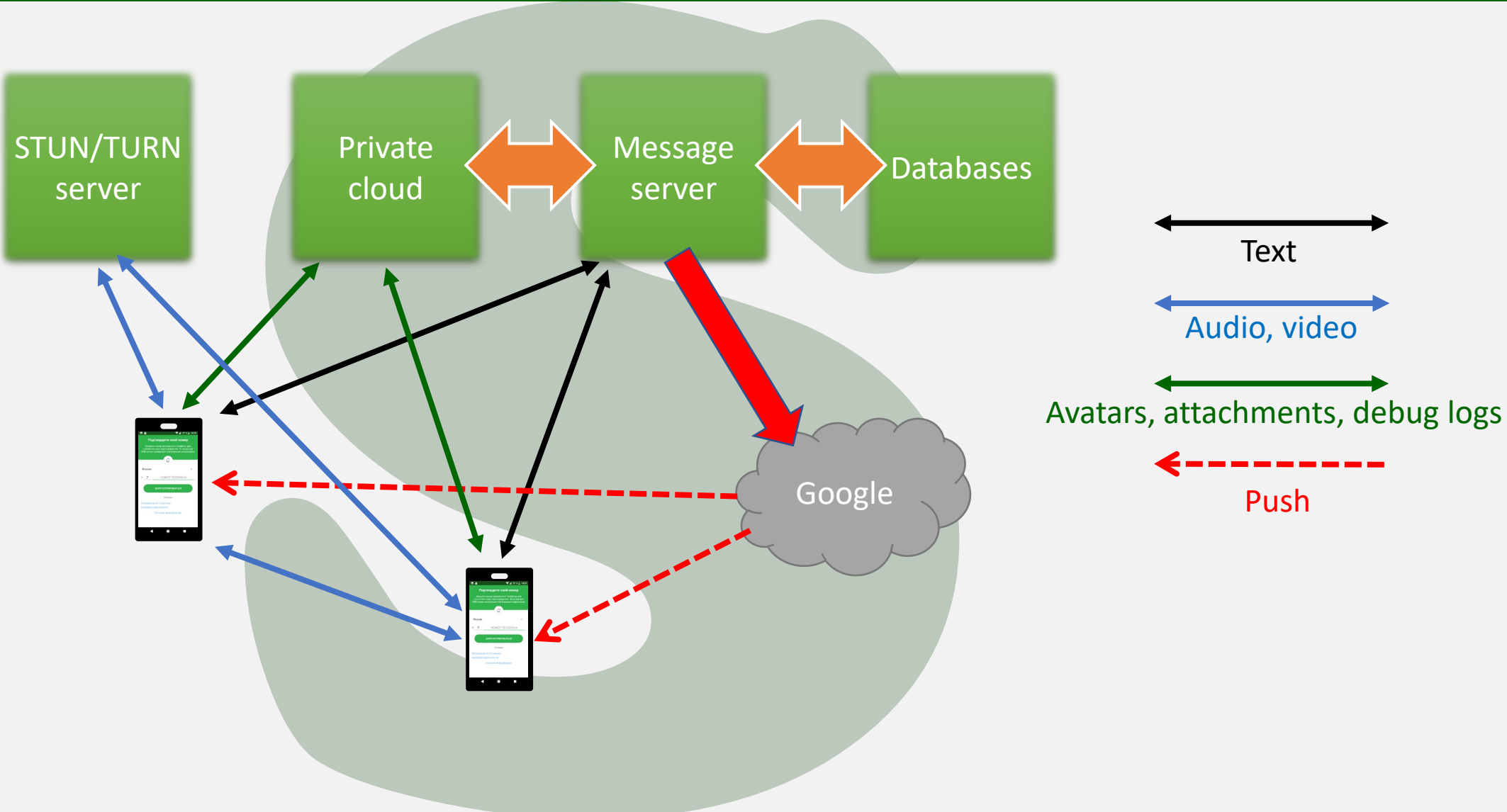
The secure messenger
Shadow
combines all three
components



Comparison of some products

Product	End-to-end encryption	Open source client	Open source server	Own server
Skype	✗	✗	✗	✗
Mattermost	✗	✓	✓	✓
WhatsApp	✓	✗	✗	✗
Telegram	✓	✓	✗	✗
ViPole	✓	✗	✗	✓
Silent Phone	✓	✓	✗	✗
Signal	✓	✓	✓	✗
Shadow	✓	✓	✓	✓

Shadow architecture



Shadow architecture

- Shadow is based on the code of Signal messenger – the world leader in the field of secure messaging
- Client and server code is completely open source
- Transmission of messages, attachments, audio and video flows, as well as user profile information, are secured with end-to-end encryption
- Your own private cloud service is part of the solution and is used for transitional storage of attachments and profiles instead of 3rd party clouds such as Amazon or Google. This information is stored only in encrypted (end-to-end) form
- The customer installs its own server which is completely under customer's administrative control. The server operates under Linux OS (CentOS, Debian...)
- The architectural components of the server are: registration & messaging server, NAT traversal server, private cloud server, database servers
- No reliance on 3rd party cloud DB services (e.g. Amazon DynamoDB). All persistent storage is on-premise

Functionality – communication modes

- Secure chat (point-to-point and group mode)
- Secure voice and video calls (point-to-point)
- Secure file transmission (images, audio/video, documents...)
- Location reporting (with positioning on map)
- Contact sharing
- End-to-end encryption of all media traffic

Functionality – metadata protection

- In contrast to certain other open source products, Shadow is aimed not only at confidentiality of communication content, but also at minimizing exposition of client metadata on the server
- The “Sealed Sender” technology allows to hide information of the message’s sender from the server
- Metadata protection is important even when the server is your own, since it could be accessed by attackers in unauthorized fashion

Functionality – account management

- In contrast to many mass market products, Shadow accounts are not linked to phone numbers. The mobile device does not need the SIM card as such
- No phone numbers -> no threats associated with SMS or callback verification
- Usernames (logins) are set by the system administrator
- The contact list is downloaded by clients from the Shadow server

Functionality – message management

- Delivery/read notifications (user-tunable)
- Automatic history cleanup (time- and quantity-based)
- Encrypted backup
- Push notifications (sound, vibration, LED)
- Typing indication (user-tunable)

Functionality – contact management

- Blacklisting unwelcome parties
- Basic profile setup (first/last name and avatar/photo)
- Selected chats shortcuts can be placed on the main screen
- Chat archiving

Functionality – security

- Protection from MITM attacks (digital fingerprint verification)
- Application lock (fingerprint or PIN)
- History backup to an encrypted file
- Automatic history cleanup (time- and quantity-based)
- Screenshot disablement
- Hiding application information when not in foreview
- Offline alarm signaling
- Forced routing of audio/video traffic through a TURN server to hide the client IP address
- Manual account management (no insecure SMS/callback registration)

Technology

- Messaging traffic, encryption of attachments and profiles:
 - Signal (X3DH, Double Ratchet)
 - Primitives: Curve 25519, AES 256, HMAC-SHA256
- Client-server signaling traffic: TLS 1.2
- Audio/video traffic: SRTP
- WebRTC, Opus **CBR**, VP8
- NAT traversal: STUN/TURN

More information at our website:

www.shadowprivacy.com

Further discussion welcome!

info@shadowprivacy.com