Homework #5
Telegram vs WhatsApp Security

CNS Course Sapienza

Riccardo Prinzivalle, 1904064

November 30, 2020

1 Homework Goal

This homework contains a basic introduction to Telegram and WhatsApp security, then a comparison of their security protocols (**MTProto** and **Signal** respectively), some past development and technical issues/vulnerabilities, and what it is possible to find about current threat and vulnerabilities.

2 Telegram Security basics

Telegram uses a security protocol called MTProto, developed by the telegram team. It is a symmetric encryption protocol based on 256-bit symmetric AES encryption, 2048-bit RSA encryption and Diffie–Hellman key exchange. The protocol is divided in 3 layers:

- **High-level** component which defines the method whereby API queries and responses are converted to binary messages.
- **Cryptographic/authorization** layer which defines the method used to encrypt messages prior to being transmitted through the transport protocol.
- **Transport** component, which defines the method for the client and the server to transmit messages over some other existing network protocol.

Let's analyze in the details every section. The high level component sees a client and a server exchanging messages inside a session, which is identified by a user key identifier (a particularity, the session is attached to the client instead of standard protocols such as http/s or tcp). The client can instantiate different connections to the server (the practicality of Telegram stands in the fact that one can open different sessions on many devices such as browsers without having to log in many times once one have the session active), and messages can be

sent from one connection to the other and everything is synchronized server side. The low level message structure can be seen in fig. 1.

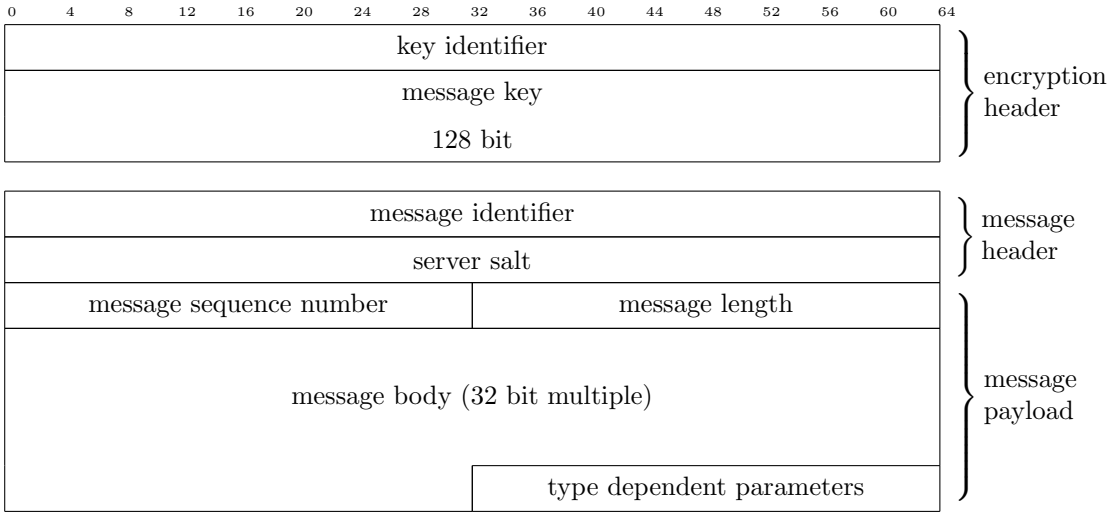

Figure 1: Telegram simplified packet structure

The message header is fixed for every message, the types of parameters does not change, and they are used by the encryption part, like message identifier and server salt.
To be mentioned, all number are saved as little endian, with the exception of large numbers, such as those needed by RSA and DH which are stored as big endian due to openSSL compatibility.
The encryption stuff can be identified in the upper part fig. 1, it is added at the end of encryption as header of the encrypted message. The message is encrypted using a 256-bit key constituted by the message key and the user key and the encryption is performed with AES-256. The message key is defined as the hash using SHA256 on the message body, and taking the 128 middle resulting bits. The user key is generated from the authorization key: it is created once, when the client is first run on the device, and never changes, so this will expose all messages if that key is stolen (even from the device or from server side); different counter measures can be taken:

- Use session keys generated at every session using the Diffie Hellman exchange protocol
- Store the keys on the device and protect them with a password
- Protect all stored and cached data of the device with a password

All these measures cannot protect the user in the case where is the server that is violated or some government agency ask the keys for terrorism prevention (as

example). The complete encryption scheme for every message can be seen in fig 2.

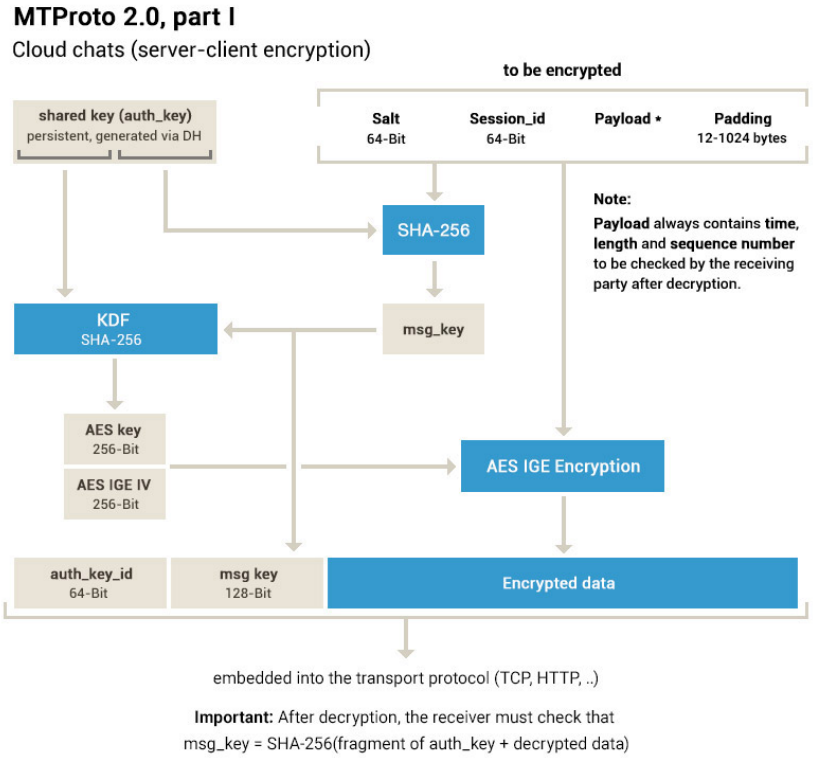

Figure 2: MTProto

The message identifier field is strictly related to time on the sender machine, and the receiver checks it with its time, so if there is a time divergence on sender or receiver message (here it is intended client/server communication and vice versa) then the receiver start to drop messages: in this case, the server (even if its time is not correct) will send a service message asking the client to synchronize its time with the server one. The time synchronization is simply defined as storing the difference between server and client time on client side. If the synchronization fails then the client must start a new session to continue the communication.
The transport layer encapsulated the encrypted message with a secondary protocol header depending on specification of the message contained in the header, this allows to obtain additional services such as *quick ack* or *transport error*. The effective transport then relies on existent transport layer protocols as TCP, websockets or HTTP/S.
This section has been developed using [1] and [25] as references.

3 WhatsApp Security Basics

WhatsApp security relies on **Signal** protocol: it allows end-to-end encryption of every communication through the application. The protocol is based on the following steps:

- Client registration and public keys exchange
- Encrypted session creation
- Message exchange using **Double Ratchet Algorithm** to generate a key for every message

When a new client is registered, some public keys are sent to the server who stores them: these keys are a *public identity key*, a *signed pre key* with its signature and a batch of *one time pre keys*. These elements allow the server to uniquely identify a user identity.
Every communication with a user is treated as an encrypted session: it is established at the first message exchanged and it does not change until no external factor changes the integrity of the session (this can happen due to a device change or an app reinstall). The session is created using the public keys of the receiver retrieved from the server and they are used to generate a master secret by "or" operation of different Diffie Hellman based on Elliptic Curves. The secret key is used by an hashed key derivation function to create a root key and some chain keys. If the recipient is not online at the time of the creation of the session, then the initiator includes the information to start the session with the first message sent, in order for the recipient to start the same encrypted session when he receives the messages.
Every message sent through a session are encrypted by a message key using AES256 and HMAC-SHA256 for authentication. The message key changes for every message and it is ephemeral, so it is not possible to reconstruct an encrypted message from the session state. The key generation is based on the **Double Ratchet Algorithm**: it is based on HMAC-SHA256 to generate the single message key from the ratchet chain key and on the association of ECDH and HKDF to generate the new chain key and root key for the next message (these steps guarantee forward secrecy). This mechanism can cause delay in the messages and shift on the order of the original messages, but since every message is encrypted separately, it is not a problem. The Double ratchet scheme can be seen in fig. 3; the letters A and B are the message sent by Alice or Bob, and the number is the number of the message sent by one of them.
The user can verify the end-to-end encryption keys to avoid a man-in-the-middle attack, the verification can be performed either by a QR code or by a 60 digit number, that the users must compare with the keys stored on their device to ensure no attack has been performed.
For what concerns the transport layer, WhatsApp uses the **Noise** protocol framework: this provides long time connections between the clients and the servers based on *noise pipes* with Curve25519, AES-GCM, and SHA256.

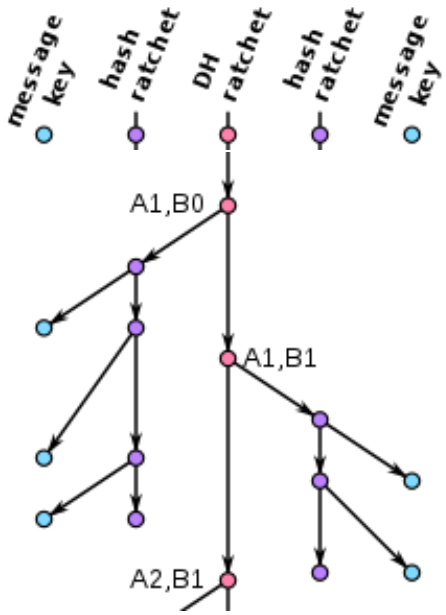

Figure 3: Double Ratchet [23]

Unfortunately, I was not able to find images like fig. 2 for WhatsApp; this is due to the nature of **Signal** protocol, which subdivides every step of security into separated applications, such as the Double Ratchet of fig. 3. This section is based on [12], [27], [13] and [2].

4 Protocols comparison

This section contains some comparisons of both protocols, evidentiating some key aspect and vulnerabilities.
It can be easily seen that telegram hides some big hole in security: the first problem is its usability, to guarantee that, telegram uses session attached to the clients, so the session decades only when the client closes it; in addition one can enter a past session which is still open in a device just by going to the site and the cookies will automatically log in the user to that session. Furthermore, everything sent with telegram (messages, media, links and files) is all stored on server side, this will facilitate the user to obtain everything from every device where he is logged in, but if it is easier for the end user also someone interested in that data will gain access in easier way. Another problem is that all messages are encrypted with a key derived from the authorization key: this key rarely changes, and exposes all messages in the cases it is stolen from a device, this is due to the structure of the key derivation, from which it is possible to decrypt every message once it is stolen.

Furthermore, not all the chats are end-to-end encrypted, only secret chats have this privilege, so a basic user does not know this difference and he generally uses the standard chats, so his message can be decrypted server side: this introduces another problem of telegram, the server keys are known to the telegram organization, and as it will be seen in the next sections, it is a big trouble for the end user. In addition to that, the server implementation is closed source, so the effectively security of server side cannot be verified by the user or others neutral entities.
Another problem is based on the time synchronization: if one of server or client have a different time, then the messages will began to be discarded, and since it is always the server who send the reference time for the synchronization, if a server is compromised, then an attacker can instantiate a DOS attack.
WhatsApp instead has better masked his security holes. This app uses end-to-end encryption for every communication, a problem stands in the initialization of the encryption: the first messages shared with a new chat/user contains the informations to start the encrypted session, so if the messages are intercepted by someone else other than the effective receiver, one can in some way listen to the session from that moment on, since the session keys changes only for rare events, such as app reinstall or device changes. An attacker can also become active if he has the session keys and inject false messages in the session. For what concerns the usability on browser, the opened session expires after some small time of inactivity.
In both cases the message key changes for every message, WhatsApp uses a more reliable message key generation, but if the session key is stolen then it is possible to rebuild every message key. If a message key is stolen, then only the associated message can be decrypted, but none of past and future message keys can be derived from another message key.
For the transport layer, Telegram uses a proprietary protocol on top of the standard ones, so transport security relies on the security of open transport layer protocols. WhatsApp instead relies completely on the Noise Protocol, which is open as in the other case of Telegram.
WhatsApp has both client and server closed source, so it has the same problem of Telegram when it comes about security verification by third party neutral entities.
The problem on both apps is that they are ruled by companies whose policies are not so secure: in one case the Telegram Messenger Inc has doubtly policies [3] and server keys are in their hand; on the contrary WhatsApp [**?**] is in the hand of Facebook, not one of the clearest companies in the world.
To recap, a small list of both protocols' problems:

- both have some pieces closed source, so no security verification
- both protocols have problems with the initialization of the session, if the session keys are stolen then all the messages can be decrypted
- telegram has also problem worth time synchronization and server side encryption (the latter only for not secret chat)

- both have problems with browser session, Telegram problem is a bit huge w.r.t. the WhatsApp counterpart

The comparison has taken in exam only the standard one-to-one chat, another analysis can comprehend also media, group chats and call/videocall security, which in the case of WhatsApp can be of big concerns [26].

5 Past Evolution and Fixed Vulnerabilities

The first app was WhatsApp, founded in 2009, at that time the app sent all the messages as plaintext, the first form of cryptography was added back in 2012, but it wasn't end-to-end-encryption and it was considered broken after few time [5]. Many securities and government entities demonstrated their doubts about WhatsApp security issues during the period from 2012 to 2015 [5], and this pushed WhatsApp to implement some better form of security. So end-to-end encryption was introduced only in 2014 [18] when WhatsApp joins Open Whisper Systems; the encryption is practically implemented using Signal protocol. Some time previous of WhatsApp e-t-e encryption was introduced, Telegram was founded in Russia (2013). The problem with Telegram is that it uses a closed source encryption protocols, which has been demonstrated to not be fully secure [20], and this protocol is still use today's. One year after Telegram foundation, their creators affirm that all the source code will eventually be released, but only the client code was released, the server side is still closed source. To overcome this problem, the Telegram company organized two cryptoanalysis contests [30] [30] to prove that its cryptography is secure enough, but suspiciously no one was able to break any vulnerabilities of the messaging app (it is generally suspected that the results are not completely clear and misleading).
In 2016 Telegram suffers an attack on the method used to authenticate users [24], the attack consists on the SMS spoofing on the messages used to authorize a new session, so it is not an attack performed against Telegram itself, but since it relies on this method for the authentication, then it is not secure. Today's it is suggested to use a 2-factor authentication to overcome this problem. During the years, many more attacks like this were perpetrated against activists and government members by exploiting SMS vulnerabilities for authentication or inserting malwares inside the device of the victims [6] [15] [7]. Another episode comprehend a big leak of Iranian Telegram accounts, together with private information, but also in this case the leak was due to problems not directly attributable to Telegram app itself, one of the method implied the exploit of Telegram desktop, but the other relied on device hacking or phishing techniques [10].
The WhatsApp e-t-e encryption needed two years to be fully implemented, and only in 2016 every form of communication made through the app is fully encrypted, and end users can verify each other keys as seen in previous sections.
To simplify friends' contact discovery, both application send to the server a copy of the mobile entire addresses' book, it is not clear how securely this data are

sent and stored on the server, but form privacy policies, it is supposed that they don't use an enough high level of security compliant with different states regulator's policies.
During the years, WhatsApp has suffered from several security bugs, such as ack crash [21] and the web app bugs [14]. It is not clear if these bugs are effectively fixed as today.
WhatsApp credibility on policies has been completely doubted after the Facebook acquisition in 2014 and later policies updated by the Facebook company.
In 2017 WhatsApp experienced a controversy with an article published on The Guardian, which discloses some vulnerabilities in e-t-e encryption, but both WhatsApp and Open Whisper Systems denied these vulnerabilities; despite these threat not being recognized officially, WhatsApp introduced a two-factor authentication and verification (this feature was already in beta, but the timing seems suspect) [5].
For further readings see Wikipedia references for both applications [25] and [13]

6 Current Vulnerabilities and Security Threats

As seen in previous section, till today's, no attack perpetrated against the Telegram application itself are known, but what is not clear is how much the Telegram organization has access to every data exchanged on their servers: many episodes of censorship has been documented [22], [8], [4], [29], [28], [9] and [11]; in different places among these, the Telegram company was pressured to disclose the server encryption key, or to loose the encryption algorithm, as happened in [19].
As of today, Telegram still has its server side source code private, this introduces doubts over thrusting the company about the effective data security, since if the server keys are accessible by the company, everyone can breach the server and obtain them or it is directly the company spying over its users, but it is not effectively documented, just hypothesis of future and current threat over our privacy.
A point in favor of WhatsApp is that both encryption protocol and messaging protocols are accessible and open source, so a technical end user or some third parties can publicly verify the security of this app, while the effectively client and server implementation are closed source, so the app is only based on these open source projects, but at the end it is a closed source application, so some vulnerabilities can be discovered slower than in open source contest.
In 2019 it was found a possibility to install a spyware through WhatsApp by making a call which did not even need answer; the company introduced a bounty hunter on the bug but as of today it seems to still be an issue [5].
As stated in [17], a new EU directive says that in the nearest future, all the European External Action Service (EEAS) will rely on a new secret communication application, not currently released instead of using WhatsApp and Signal: this brings up new problems, such as the open sourceness of the project (probably not), and consequently the real level of security provided by this new app (as

already know, security by obfuscation is never a good idea); at least the data EU diplomacy chat will remains in the hand of a government agency and not in some extra company whose policies are not that clear.
Other threats can arise from metadata [16]: this article may seem referring to Telegram only, instead it introduces also WhatsApp, Signal and iMessage vulnerabilities caused by metadata which remains on the sender device and sometimes are sent in clear to the server as information on identification and for other purposes. This exposes another WhatsApp problem (and of many other instant messaging applications): the chat backups are clearly stored on the user device, so if it is compromised, at the end of the day an attacker can easily access all the chat present in the backup.
Even if the application are "secure" from computer point of view, there is always the human factor which weakens all security levels: a possible threat is to try to access the user's account and then steal the two-factor code by social engineering techniques, as stated in [**?**].

7 Conclusion

After this brief introduction on elliptic curves, it is obvious why they have been widely adopted in many cases of asymmetric encryption: they use less bits for the same level of security, so are more efficient to compute and their base concepts are easier to visualize. As suggested on section , EC can be easily used in hybrid encryption scheme, in the key exchange phase.

References

[1] Mtproto mobile protocol. `https://core.telegram.org/mtproto`.

[2] Signal technical information. `https://signal.org/docs/`.

[3] Telegram privacy policy. `https://telegram.org/privacy`.

[4] leading-bahraini-isps-are-blocking-telegram-traffic. `https://bahrainwatch.org/blog/2016/06/28/leading-bahraini-isps-are-blocking-telegram-traffic/`, 2016.

[5] Reception and criticism of whatsapp security and privacy features. `https://en.wikipedia.org/wiki/Reception_and_criticism_of_WhatsApp_security_and_privacy_features`, 2019.

[6] Secret brazil archive. `https://theintercept.com/series/secret-brazil-archive/`, 2019.

[7] Telegramgate wikipedia page. `https://en.wikipedia.org/wiki/Telegramgate`, 2019.

[8] Urgent statement of hkispa on selective blocking of internet services. https://www.hkispa.org.hk/139-urgent-statement-of-hkispa-on-selective-blocking-of-internet-services.html, 2019.

[9] Blocking telegram in russia. https://en.wikipedia.org/wiki/Blocking_Telegram_in_Russia, 2020.

[10] Check point researchers: Iranian hackers can bypass encrypted apps like telegram. https://www.securitymagazine.com/articles/93423-check-point-researchers-Iranian-hackers-can-bypass-encrypted-apps-like-telegram, 2020.

[11] Telegram in iran. https://en.wikipedia.org/wiki/Telegram_in_Iran, 2020.

[12] Whatsapp encryption overview. https://scontent.whatsapp.net/v/t39.8562-34/122249142_469857720642275_2152527586907531259_n.pdf/WA_Security_WhitePaper.pdf?ccb=2&_nc_sid=2fbf2a&_nc_ohc=zyXCDnfHxJEAX_DdSo3&_nc_ht=scontent.whatsapp.net&oh=44e5aa8afd1092357784e47a197b5b32&oe=5FD7D019, 2020.

[13] Whatsapp wikipedia page. https://en.wikipedia.org/wiki/WhatsApp#Technical, 2020.

[14] I. Bhuyan. Multiple vulneribilities found in whatsapp web. https://www.hackatrick.com/2015/02/multiple-vulneribilities-found-in.html, 2015.

[15] T. Brewster. Mystery russian telegram hacks intercept secret codes to spy on messages. https://www.forbes.com/sites/thomasbrewster/2019/12/12/mystery-russian-telegram-hacks-intercept-secret-codes-to-spy-on-messages/, 2019.

[16] K. Chiu. Why telegram isn't as secure as you think. https://www.scmp.com/abacus/tech/article/3029415/why-telegram-isnt-secure-you-think, 2019.

[17] Z. Doffman. Whatsapp and signal replaced by new mystery messaging app: This eu change matters—here's why. https://www.forbes.com/sites/zakdoffman/2020/02/27/whatsapp-and-signal-replaced-by-new-mystery-messaging-app-this-eu-change-matters-heres-why/#5c82e0677ba9, 2020.

[18] J. Evans. Whatsapp partners with open whispersystems to end-to-end encrypt billions of messages a day. https://techcrunch.com/2014/11/18/end-to-end-for-everyone/?guccounter=1&guce_referrer=aHR0cHM6Ly9lbi53aWtpcGVkaWEub3JnLw&guce_referrer_sig=AQAAABascdn-ZgNbPnrTm2wQQtPv9HOQAwaghdjZXqBwdrr52lEJzbnV2nT7FYbWMDrMCMQzKD8gPdV-Sm4HaEHxiLwvpX5DyTUyXIai1fZbspDejl23URUUOO5eZeTJE1MLG3cUrYOLQbTn4nrIXnTiv-MRTeBI2xOLOArWg7ZIVzvB, 2014.

[19] M. Grothaus. Telegram founder: U.s. intelligence agencies tried to bribe us to weaken encryption. https://www.fastcompany.com/4040876/telegram-founder-u-s-intelligence-agencies-tried-to-bride-him-to-weaken-encryption, 2017.

[20] J. Jakobsen and C. Orlandi. On the cca (in)security of mtproto. https://eprint.iacr.org/2015/1177.pdf, 2015.

[21] S. Khandelwal. Crash your friends' whatsapp remotely with just a message. https://thehackernews.com/2014/12/crash-your-friends-whatsapp-remotely_1.html, 2014.

[22] R. Kilpatrick. China blocks telegram messenger, blamed for aiding human rights lawyers. https://hongkongfp.com/2015/07/13/china-blocks-telegram-messenger-blamed-for-aiding-human-rights-lawyers/, 2015.

[23] P. kitty111. https://en.wikipedia.org/wiki/Double_Ratchet_Algorithm#/media/File:Axolotl_ratchet_scheme.svg.

[24] T. Lokot. Is telegram really safe for activists under threat? these two russians aren't so sure. https://advox.globalvoices.org/2016/05/02/is-telegram-really-safe-for-activists-under-threat-these-two-russians-arent-so-sure/, 2016.

[25] Multiple. Telegram wikipedia page. https://en.wikipedia.org/wiki/Telegram_(software), 2020.

[26] K. O'Flaherty. Whatsapp security: Is this hidden flaw a new reason to quit? https://www.forbes.com/sites/kateoflahertyuk/2020/02/29/whatsapp-security-is-this-hidden-flaw-a-new-reason-to-leave/, 2020.

[27] A. Panghal. Whatsapp's end to end encryption, how does it work? https://medium.com/@panghalamit/whatsapp-s-end-to-end-encryption-how-does-it-work-80020977caa0, 2018.

[28] S. S. Ravikumar. Reddit, telegram among websites blocked in india: internet groups. https://in.reuters.com/article/us-india-internet/reddit-telegram-among-websites-blocked-in-india-internet-groups-idINKCN1RF14D, 2019.

[29] P. Staff. Telegram blocked in pakistan! https://propakistani.pk/2017/11/17/telegram-blocked-pakistan/, 2017.

[30] T. T. Team. Winter contest ends. https://telegram.org/blog/winter-contest-ends, 2014.