Subverting Telegram's End-to-End Encryption

Benoît Cogliati, Jordan Ethan, Ashwin Jha

CISPA Helmholtz Center for Information Security, Saarbrücken, Germany

March 22, 2023

Table of Contents

Why Telegram?

- Over 500 million active users.

Why Telegram?

- Over 500 million active users.
- Claims to be faster and safer than alternatives.

Why Telegram?

- Over 500 million active users.
- Claims to be faster and safer than alternatives.
- Uses it's own security protocol - MTProto.

Why Telegram?

- Over 500 million active users.
- Claims to be faster and safer than alternatives.
- Uses it's own security protocol - MTProto.
- Not a lot of existing analysis.

Conversation Modes

- Cloud chats:
  - Uses client $\Leftrightarrow$ server encryption.
  - Messages are stored (encrypted) in the server (Telegram's).
  - Chat history accessible across devices.

Conversation Modes

- Cloud chats:
  - Uses client $\Leftrightarrow$ server encryption.
  - Messages are stored (encrypted) in the server (Telegram's).
  - Chat history accessible across devices.
- Secret chats:
  - Uses E2EE.
  - Messages are stored locally only.

MTProto security

- MTProto 1.0 not IND-CCA - Jakobsen, and Orlandi in 2016 [JO16].

MTProto security

- MTProto 1.0 not IND-CCA - Jakobsen, and Orlandi in 2016 [JO16].
- Telegram revised scheme - MTProto 2.0 (IND-CCA?)

MTProto security

- MTProto 1.0 not IND-CCA - Jakobsen, and Orlandi in 2016 [JO16].
- Telegram revised scheme - MTProto 2.0 (IND-CCA?)
- MTProto 2.0 is IND-CCA - Albrecht et al., 2022 [Alb+22] (requires non-standard assumptions on building blocks).

MTProto security

- MTProto 1.0 not IND-CCA - Jakobsen, and Orlandi in 2016 [JO16].
- Telegram revised scheme - MTProto 2.0 (IND-CCA?)
- MTProto 2.0 is IND-CCA - Albrecht et al., 2022 [Alb+22] (requires non-standard assumptions on building blocks).
- What about practical attacks? subversion attacks?

Subversion Attacks - Motivation

- Snowden revelations - mass surveillance of the internet is taking place by governmental agencies.

Subversion Attacks - Motivation

- Snowden revelations - mass surveillance of the internet is taking place by governmental agencies.
- Is just encrypting the data enough? No

Subversion Attacks - Motivation

- Snowden revelations - mass surveillance of the internet is taking place by governmental agencies.
- Is just encrypting the data enough? No
- Agencies can inject backdoors into secure implementations by manipulating the encryption algorithms.

Subversion Attacks - History

- Foundations by Young and Yung in 90's [YY96; YY97] (kleptography).
  - Output of subverted algorithm is computationally indistinguishable from output of unmodified algorithm.
  - Subverted algorithm should leak the secret key through the output.

Subversion Attacks - History

- Foundations by Young and Yung in 90's [YY96; YY97] (kleptography).
  - Output of subverted algorithm is computationally indistinguishable from output of unmodified algorithm.
  - Subverted algorithm should leak the secret key through the output.
- Later on Bellare et al. in 2014 [BPR14], introduced Algorithm Substitution Attacks (ASAs) against randomized encryption schemes.
  - Relies on randomness generated in the course of encryption.
  - Attack works against sub-class of randomized schemes (coin-injectivity).

Our Contribution

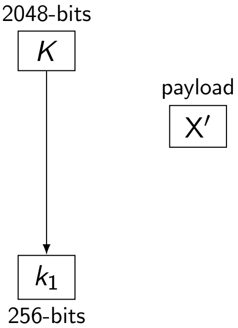
1. First partial key recovery algorithm substitution attack (ASA) on secret chat mode of Telegram:
    - exploit the random padding (and length) used during the encryption.
    - our attack works on desktop client and tdlib library (used by third party clients).

Our Contribution

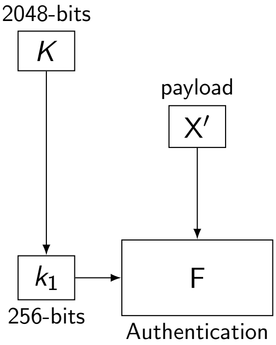
1. First partial key recovery algorithm substitution attack (ASA) on secret chat mode of Telegram:
   - exploit the random padding (and length) used during the encryption.
   - our attack works on desktop client and tdlib library (used by third party clients).
2. The subversion attack can be averted (modified version).

Table of Contents

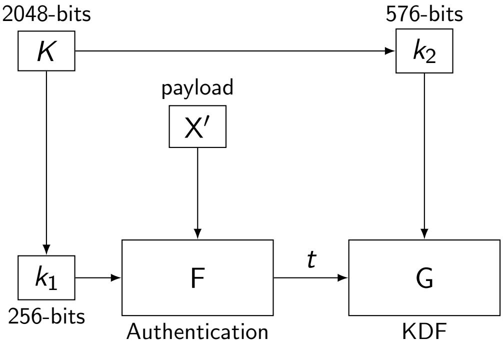
MTProto 2.0 - Scheme



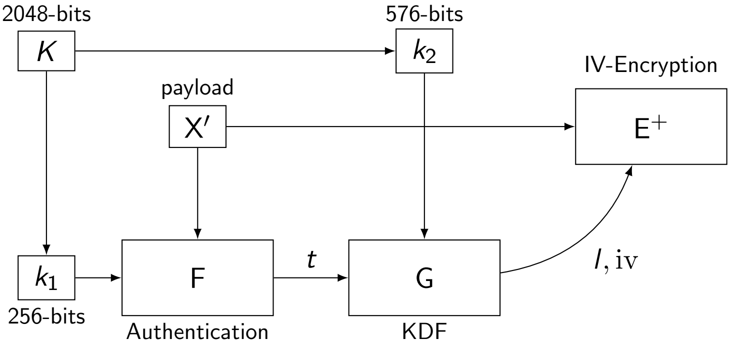
MTProto 2.0 - Scheme



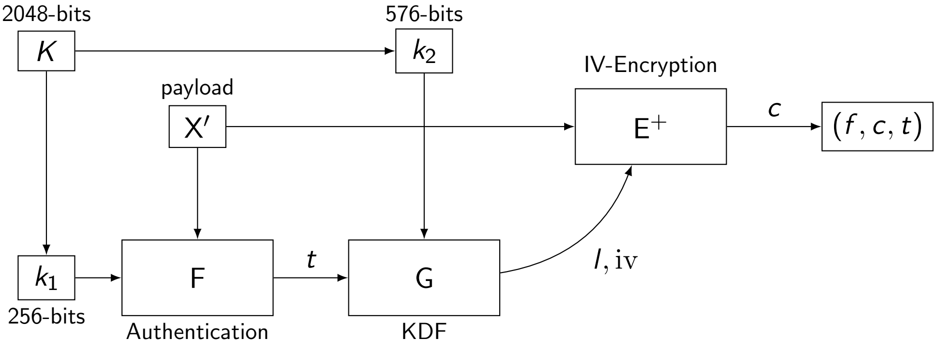
MTProto 2.0 - Scheme



MTProto 2.0 - Scheme



MTProto 2.0 - Scheme



Payload for Secret Chats

- The Full message is defined by:

$$X := \textbf{length} \| \textbf{payload\_type} \| \textbf{random\_bytes} \| \textbf{layer} \| \textbf{in\_seq\_no} \\ \| \textbf{out\_seq\_no} \| \textbf{message\_type} \| \textbf{message\_data}$$

Payload for Secret Chats

- The Full message is defined by:

$$X := \textbf{length} \| \textbf{payload\_type} \| \textbf{random\_bytes} \| \textbf{layer} \| \textbf{in\_seq\_no} \\ \| \textbf{out\_seq\_no} \| \textbf{message\_type} \| \textbf{message\_data}$$

- The payload is defined by:

$$X' := X \| \textbf{random\_padding}$$

Payload for Secret Chats

- The Full message is defined by:

$$X := \textbf{length} \| \textbf{payload\_type} \| \textbf{random\_bytes} \| \textbf{layer} \| \textbf{in \_seq\_no} \\ \| \textbf{out\_seq\_no} \| \textbf{message\_type} \| \textbf{message\_data}$$

- The payload is defined by:

$$X' := X \| \textbf{random\_padding}$$

- In our attack we will use three of these fields:

Payload for Secret Chats

- The Full message is defined by:

$$X := \textbf{length} \| \textbf{payload\_type} \| \textbf{random\_bytes} \| \textbf{layer} \| \textbf{in \_seq\_no} \\ \| \textbf{out\_seq\_no} \| \textbf{message\_type} \| \textbf{message\_data}$$

- The payload is defined by:

$$X' := X \| \textbf{random\_padding}$$

- In our attack we will use three of these fields:
  - **in _seq_no** + **out _seq_no** $\rightarrow$ used to derive the state for our attack.

Payload for Secret Chats

- The Full message is defined by:

$$\begin{aligned} \mathrm{X} := \textbf{length} \| \textbf{payload\_type} \| \textbf{random\_bytes} \| \textbf{layer} \| \textbf{in\_seq\_no} \\ \| \textbf{out\_seq\_no} \| \textbf{message\_type} \| \textbf{message\_data} \end{aligned}$$

- The payload is defined by:

$$\mathrm{X}' := \mathrm{X} \| \textbf{random\_padding}$$

- In our attack we will use three of these fields:
  - **in _seq_no** + **out _seq_no** → used to derive the state for our attack.
  - $\sigma' = |\mathrm{X}'| =$ length is random and divisible by 16 bytes → the main vulnerability for our attack.

Payload for Secret Chats

- The Full message is defined by:

$$\mathsf{X} := \textbf{length} \| \textbf{payload\_type} \| \textbf{random\_bytes} \| \textbf{layer} \| \textcolor{blue}{\textbf{in\_seq\_no}} \\ \| \textcolor{blue}{\textbf{out\_seq\_no}} \| \textbf{message\_type} \| \textbf{message\_data}$$

- The payload is defined by:

$$\mathsf{X}' := \mathsf{X} \| \textcolor{red}{\textbf{random\_padding}}$$

- In our attack we will use three of these fields:
  - **in _seq_no** + **out _seq_no** → used to derive the state for our attack.
  - $\sigma' = |\mathsf{X}'|$ = length is random and divisible by 16 bytes → the main vulnerability for our attack.
  - **random_padding** = 12 to 1024 random bytes → used to improve our attack.

Sampling Padding Length

- Upon code inspection $\rightarrow$ padding length is sampled differently for each platform.

Sampling Padding Length

- Upon code inspection $\rightarrow$ padding length is sampled differently for each platform.
- We concentrate on: desktop client & tdlib libary (for third party).

Sampling Padding Length

- Upon code inspection $\rightarrow$ padding length is sampled differently for each platform.
- We concentrate on: desktop client & tdlib libary (for third party).
- **Original Sampling:** (Message of σ bytes)

$$\sigma' = g(\sigma) + v, v \leftarrow\!\!\$ \, [0, 15]. \qquad \text{(16-byte block)}$$

Sampling Padding Length

- Upon code inspection $\rightarrow$ padding length is sampled differently for each platform.
- We concentrate on: desktop client & tdlib libary (for third party).
- **Original Sampling:** (Message of σ bytes)

$$\sigma' = g(\sigma) + v, v \leftarrow\$ [0, 15]. \qquad \text{(16-byte block)}$$

- **Alternative Sampling:**

$$\sigma' \mod 16 \leftarrow\$ [0, 15] \rightarrow \sigma' : \lfloor(\sigma' - g(\sigma))/16\rfloor > 0. \qquad \text{(16-byte block)}$$

Sampling Padding Length

- Upon code inspection $\rightarrow$ padding length is sampled differently for each platform.
- We concentrate on: desktop client & tdlib libary (for third party).
- **Original Sampling:** (Message of σ bytes)

$$\sigma' = g(\sigma) + v, v \leftarrow\!\$\ [0, 15]. \qquad \text{(16-byte block)}$$

- **Alternative Sampling:**

$$\sigma' \mod 16 \leftarrow\!\$\ [0, 15] \rightarrow \sigma' : \lfloor (\sigma' - g(\sigma))/16 \rfloor > 0. \qquad \text{(16-byte block)}$$

- This second sampling mechanism will prove useful for our attack.

Sampling Padding Length

- Upon code inspection $\rightarrow$ padding length is sampled differently for each platform.
- We concentrate on: desktop client & tdlib libary (for third party).
- **Original Sampling:** (Message of σ bytes)

$$\sigma' = g(\sigma) + v, v \leftarrow\!\$ \, [0, 15]. \qquad \text{(16-byte block)}$$

- **Alternative Sampling:**

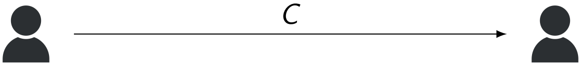
$$\sigma' \mod 16 \leftarrow\!\$ \, [0, 15] \rightarrow \sigma' : \lfloor (\sigma' - g(\sigma))/16 \rfloor > 0. \qquad \text{(16-byte block)}$$

- This second sampling mechanism will prove useful for our attack.
- $Pad(M, v) = M$ is padded to a message whose length in 16-byte length is $v \mod 16$.

Table of Contents

Algorithm Substitution Attacks

Normal Setting:

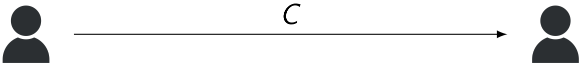


$C = \mathcal{E}^{+}(K_E, A, M, \mathrm{iv}, \mathit{state})$

$M = \mathcal{E}^{-}(K_E, A, C, \mathit{state}')$

Subverted Setting:

Normal Setting:



$C = \mathcal{E}^{+}(K_E, A, M, \mathrm{iv}, \mathit{state})$

$M = \mathcal{E}^{-}(K_E, A, C, \mathit{state}')$

Subverted Setting:

$C = \widetilde{\mathcal{E}^{+}}(K_A, K_E, A, M, \mathrm{iv}, \mathit{state})$

$M = \mathcal{E}^{-}(K_E, A, C, \mathit{state}')$ (Decryptable)

$$\downarrow \widetilde{\mathcal{E}^{\mathrm{ext}}}(\mathbf{A}, \mathbf{C})$$

$(K = K_E)?$

Algorithm Substitution Attacks - Goals

- The subversion has two goals:

Algorithm Substitution Attacks - Goals

- The subversion has two goals:
  - **Undetectability:** as long as client/server does not have access to K_A, the outputs of the real encryption and the subverted encryption are indistinguishable.

Algorithm Substitution Attacks - Goals

- The subversion has two goals:
  - **Undetectability:** as long as client/server does not have access to K_A, the outputs of the real encryption and the subverted encryption are indistinguishable.
  - **Key Recovery:** the subversion is able to recover a part of the key K_E.

Algorithm Substitution Attacks - Goals

- The subversion has two goals:
  - **Undetectability:** as long as client/server does not have access to K_A, the outputs of the real encryption and the subverted encryption are indistinguishable.
  - **Key Recovery:** the subversion is able to recover a part of the key K_E.
- If no state was used it is called stateless (otherwise stateful).

Attack - Key Ideas

- Our attack relies on the same idea as the one from Bellare et al. [BJK15].

Attack - Key Ideas

- Our attack relies on the same idea as the one from Bellare et al. [BJK15].
- They present a very simple subversion attack for randomized IV-based encryption schemes that relies on a PRF F with output space of size $n + 1$ bits ($|K_E| = n$).

Attack - Key Ideas

- Our attack relies on the same idea as the one from Bellare et al. [BJK15].
- They present a very simple subversion attack for randomized IV-based encryption schemes that relies on a PRF F with output space of size $n+1$ bits ($|K_E| = n$).
- The idea of the attack:
  - **Subversion:** Sample iv until the cipher text $C = \mathcal{E}(K_E, A, M, \mathrm{iv})$ satisfies $F_{K_A}(C) = (b, i)$ where $K_E[i] = b$ (or s rounds have passed).
  - **Key Recovery:** Recover the key bits from the ciphertexts returned by the subversion.

Attack - Key Ideas

- Our attack relies on the same idea as the one from Bellare et al. [BJK15].
- They present a very simple subversion attack for randomized IV-based encryption schemes that relies on a PRF F with output space of size $n+1$ bits ($|K_E| = n$).
- The idea of the attack:
  - **Subversion:** Sample iv until the cipher text $C = \mathcal{E}(K_E, A, M, \mathrm{iv})$ satisfies $F_{K_A}(C) = (b, i)$ where $K_E[i] = b$ (or s rounds have passed).
  - **Key Recovery:** Recover the key bits from the ciphertexts returned by the subversion.
- The attack is stateless (to avoid state rests) but it can fail on a specific key bit.

Our Scenario

- Our setting differs from Bellare et al.[BJK15], in two main points:

Our Scenario

- Our setting differs from Bellare et al.[BJK15], in two main points:
  - MTProto 2.0 maintains a state for each key $\rightarrow$ subversion can be stateful.

Our Scenario

- Our setting differs from Bellare et al.[BJK15], in two main points:
    - MTProto 2.0 maintains a state for each key $\rightarrow$ subversion can be stateful.
    - The communicating parties are honest and interested in secure communication $\rightarrow$ the key is assumed to be generated at random ([AP19]).

Our Scenario

- Our setting differs from Bellare et al.[BJK15], in two main points:
  - MTProto 2.0 maintains a state for each key $\rightarrow$ subversion can be stateful.
  - The communicating parties are honest and interested in secure communication $\rightarrow$ the key is assumed to be generated at random ([AP19]).
- We present two subversion attacks on MTProto 2.0.

Our Scenario

- Our setting differs from Bellare et al.[BJK15], in two main points:
  - MTProto 2.0 maintains a state for each key $\rightarrow$ subversion can be stateful.
  - The communicating parties are honest and interested in secure communication $\rightarrow$ the key is assumed to be generated at random ([AP19]).
- We present two subversion attacks on MTProto 2.0.
- The first attack relies on an additional length-preserving deterministic encryption scheme E.

Attack 1

- **Subversion:** $\widetilde{\mathcal{E}^{+}}(K_A, K_E, M, \sigma)$

Attack 1

- **Subversion:** $\widetilde{\mathcal{E}^{+}}(K_A, K_E, M, \sigma)$
    - Encrypt $Y = E_{K_A}(K_E)$.

Attack 1

- **Subversion:** $\widetilde{\mathcal{E}^{+}}(K_A, K_E, M, \sigma)$
  - Encrypt $Y = E_{K_A}(K_E)$.
  - Derive padding length $\mathrm{len} \leftarrow Y[4\sigma, 4\sigma + 3]$ from the state $\sigma < |K_E|/4$.

Attack 1

- **Subversion:** $\widetilde{\mathcal{E}^+}(K_A, K_E, M, \sigma)$
  - Encrypt $Y = E_{K_A}(K_E)$.
  - Derive padding length $\mathrm{len} \leftarrow Y[4\sigma, 4\sigma + 3]$ from the state $\sigma < |K_E|/4$.
  - Pad the message $M \leftarrow \mathsf{pad}(M, \mathrm{len})$.

Attack 1

- **Subversion:** $\widetilde{\mathcal{E}^+}(K_A, K_E, M, \sigma)$
    - Encrypt $Y = E_{K_A}(K_E)$.
    - Derive padding length $\mathrm{len} \leftarrow Y[4\sigma, 4\sigma + 3]$ from the state $\sigma < |K_E|/4$.
    - Pad the message $M \leftarrow \mathsf{pad}(M, \mathrm{len})$.
    - Encrypt $(C, T) \leftarrow \mathcal{E}^+(K_E, M)$

Attack 1

- **Subversion:** $\widetilde{\mathcal{E}^+}(K_A, K_E, M, \sigma)$
  - Encrypt $Y = E_{K_A}(K_E)$.
  - Derive padding length $\mathrm{len} \leftarrow Y[4\sigma, 4\sigma + 3]$ from the state $\sigma < |K_E|/4$.
  - Pad the message $M \leftarrow \mathsf{pad}(M, \mathrm{len})$.
  - Encrypt $(C, T) \leftarrow \mathcal{E}^+(K_E, M)$
- **Key Recovery:** $\widetilde{\mathcal{E}^{\mathsf{ext}}}(K_A, \mathbf{C}, \sigma)$

Attack 1

- **Subversion:** $\widetilde{\mathcal{E}^{+}}(K_A, K_E, M, \sigma)$
  - Encrypt $Y = E_{K_A}(K_E)$.
  - Derive padding length $\mathrm{len} \leftarrow Y[4\sigma, 4\sigma + 3]$ from the state $\sigma < |K_E|/4$.
  - Pad the message $M \leftarrow \mathsf{pad}(M, \mathrm{len})$.
  - Encrypt $(C, T) \leftarrow \mathcal{E}^{+}(K_E, M)$
- **Key Recovery:** $\widetilde{\mathcal{E}^{\mathsf{ext}}}(K_A, \mathbf{C}, \sigma)$
  - For every ciphertext $C \in \mathbf{C}$, $C_v := (C/16) \mod 16$.

Attack 1

- **Subversion:** $\widetilde{\mathcal{E}^{+}}(K_A, K_E, M, \sigma)$
  - Encrypt $Y = E_{K_A}(K_E)$.
  - Derive padding length $\mathrm{len} \leftarrow Y[4\sigma, 4\sigma + 3]$ from the state $\sigma < |K_E|/4$.
  - Pad the message $M \leftarrow \mathsf{pad}(M, \mathrm{len})$.
  - Encrypt $(C, T) \leftarrow \mathcal{E}^{+}(K_E, M)$
- **Key Recovery:** $\widetilde{\mathcal{E}^{\mathsf{ext}}}(K_A, \mathbf{C}, \sigma)$
  - For every ciphertext $C \in \mathbf{C}$, $C_v := (C/16) \mod 16$.
  - Join all values C_v together and form Y of size $|K_E| = n$.

Attack 1

- **Subversion:** $\widetilde{\mathcal{E}^{+}}(K_A, K_E, M, \sigma)$
  - Encrypt $Y = E_{K_A}(K_E)$.
  - Derive padding length $\mathrm{len} \leftarrow Y[4\sigma, 4\sigma + 3]$ from the state $\sigma < |K_E|/4$.
  - Pad the message $M \leftarrow \mathsf{pad}(M, \mathrm{len})$.
  - Encrypt $(C, T) \leftarrow \mathcal{E}^{+}(K_E, M)$
- **Key Recovery:** $\widetilde{\mathcal{E}^{\mathsf{ext}}}(K_A, \mathbf{C}, \sigma)$
  - For every ciphertext $C \in \mathbf{C}$, $C_v := (C/16) \mod 16$.
  - Join all values C_v together and form Y of size $|K_E| = n$.
  - Decrypt $K = E^{-}_{K_A}(Y)$ to obtain key guess.

Attack 1 - Analysis

- $\mathcal{E}$ is indistinguishable from $\tilde{\mathcal{E}}$ as long as E is secure.

Attack 1 - Analysis

- $\mathcal{E}$ is indistinguishable from $\tilde{\mathcal{E}}$ as long as E is secure.
- If $(q > |K|/4)$ the key recovery is successful.

Attack 1 - Analysis

- $\mathcal{E}$ is indistinguishable from $\tilde{\mathcal{E}}$ as long as E is secure.
- If $(q > |K|/4)$ the key recovery is successful.
- Maximum of key bits recovered $\leq 100 \cdot 4$ (because of key regeneration).

Attack 1 - Analysis

- $\mathcal{E}$ is indistinguishable from $\tilde{\mathcal{E}}$ as long as E is secure.
- If $(q > |K|/4)$ the key recovery is successful.
- Maximum of key bits recovered $\leq 100 \cdot 4$ (because of key regeneration).
- The adversary is interested in reading messages $\rightarrow$ needs 576-bit of the master key for the encryption pass.

Attack 1 - Analysis

- $\mathcal{E}$ is indistinguishable from $\tilde{\mathcal{E}}$ as long as E is secure.
- If $(q > |K|/4)$ the key recovery is successful.
- Maximum of key bits recovered $\leq 100 \cdot 4$ (because of key regeneration).
- The adversary is interested in reading messages $\rightarrow$ needs 576-bit of the master key for the encryption pass.
- Key size is $576 > 400$-bits, luckily padding bits have not been exploited!

Attack 1 - Analysis

- $\mathcal{E}$ is indistinguishable from $\tilde{\mathcal{E}}$ as long as E is secure.
- If $(q > |K|/4)$ the key recovery is successful.
- Maximum of key bits recovered $\leq 100 \cdot 4$ (because of key regeneration).
- The adversary is interested in reading messages $\rightarrow$ needs 576-bit of the master key for the encryption pass.
- Key size is $576 > 400$-bits, luckily padding bits have not been exploited!
- We present an improved algorithm, that uses an additional PRF F with output space of δ bits and adversary key K'_A, $(\tau = (4 + \delta)\sigma)$.

Attack 2

- **Subversion:** $\widetilde{\mathcal{E}^{+}}(K_A, K_E, M, \sigma)$
    - Encrypt $Y = E_{K_A}(K_E)$.
    - Derive padding length $\mathrm{len} \leftarrow Y[4\sigma, 4\sigma + 3]$ from the state $\sigma < |K_E|/4$.
    - Pad the message $M \leftarrow \mathsf{pad}(M, \mathrm{len})$.
    - Encryption:

- **Key Recovery:** $\widetilde{\mathcal{E}^{\mathsf{ext}}}(K_A, \mathbf{C}, \sigma)$
    - For every ciphertext $C \in \mathbf{C}$, $C_v := (C/16) \mod 16$.
    - Join all values C_v together and form Y of size $|K_E| = n$.
    - Decrypt $K = E^{-}_{K_A}(Y)$ to obtain key guess.

Attack 2

- **Subversion:** $\widetilde{\mathcal{E}^{+}}(K_A, K_E, M, \sigma)$
  - Encrypt $Y = E_{K_A}(K_E)$.
  - Derive padding length $\mathrm{len} \leftarrow Y[4\sigma, 4\sigma + 3]$ from the state $\sigma < |K_E|/4$.
  - Pad the message $M \leftarrow \mathsf{pad}(M, \mathrm{len})$.
  - Encryption:
    - Compute tag $T \leftarrow \mathsf{F}(K_E, pad(M, len))$ until $F_{K'_A}(T) = Y[\tau + 3, \tau + 3 + \delta]$ (or s rounds have passed).

- **Key Recovery:** $\widetilde{\mathcal{E}^{\mathsf{ext}}}(K_A, \mathbf{C}, \sigma)$
  - For every ciphertext $C \in \mathbf{C}$, $C_v := (C/16) \mod 16$.
  - Join all values C_v together and form Y of size $|K_E| = n$.
  - Decrypt $K = E^{-}_{K_A}(Y)$ to obtain key guess.

Attack 2

- **Subversion:** $\widetilde{\mathcal{E}^{+}}(K_A, K_E, M, \sigma)$
  - Encrypt $Y = E_{K_A}(K_E)$.
  - Derive padding length $\mathrm{len} \leftarrow Y[4\sigma, 4\sigma + 3]$ from the state $\sigma < |K_E|/4$.
  - Pad the message $M \leftarrow \mathsf{pad}(M, \mathrm{len})$.
  - Encryption:
    - Compute tag $T \leftarrow \mathsf{F}(K_E, pad(M, len))$ until $F_{K_A'}(T) = Y[\tau + 3, \tau + 3 + \delta]$ (or s rounds have passed).
    - Continue the original encryption scheme.
- **Key Recovery:** $\widetilde{\mathcal{E}^{\mathsf{ext}}}(K_A, \mathbf{C}, \sigma)$
  - For every ciphertext $C \in \mathbf{C}$, $C_v := (C/16) \mod 16$.
  - Join all values C_v together and form Y of size $|K_E| = n$.
  - Decrypt $K = E^{-}_{K_A}(Y)$ to obtain key guess.

Attack 2 - Analysis

- $\mathsf{G}, \mathsf{E}^{+}$ (heavy computation) computed only once instead of s times (reduces cost).

Attack 2 - Analysis

- G, E^+ (heavy computation) computed only once instead of s times (reduces cost).
- F is an iterated hash $\rightarrow$ save computation on fixed blocks and hash only random padding (saves computation each loop iteration).

Attack 2 - Analysis

- G, E^+ (heavy computation) computed only once instead of s times (reduces cost).
- F is an iterated hash $\rightarrow$ save computation on fixed blocks and hash only random padding (saves computation each loop iteration).
- **Undetectability:** For some parameters and as long as the advantages for the adversaries of PRF F, encryption scheme E as well as of Mtproto $\mathcal{E}$ are all negligible then so is the advantage for our detectability adversary.

Attack 2 - Analysis

- G, E^+ (heavy computation) computed only once instead of s times (reduces cost).
- F is an iterated hash $\rightarrow$ save computation on fixed blocks and hash only random padding (saves computation each loop iteration).
- **Undetectability:** For some parameters and as long as the advantages for the adversaries of PRF F, encryption scheme E as well as of Mtproto $\mathcal{E}$ are all negligible then so is the advantage for our detectability adversary.
- **Key Recovery:** For $q \geq \lceil |K_E|/(4+\delta) \rceil$, the same parameters as above and as long as the advantages for the adversaries of PRF F and Mtproto $\mathcal{E}$ are negligible then the key recovery success is at least $\approx 1 - qe^{-\delta s}$.

Success of Our Attack

δ	s	q	k	Pr
2	21	50	300	≥ 0.88
4	91	50	400	≥ 0.85
6	369	50	500	≥ 0.85
8	1485	50	600	≥ 0.85
10	5946	50	700	≥ 0.85

- Dominating term of success is $qe^{-\delta s}$ for $q \leq 100$.

Success of Our Attack

δ	s	q	k	Pr
2	21	50	300	≥ 0.88
4	91	50	400	≥ 0.85
6	369	50	500	≥ 0.85
8	1485	50	600	≥ 0.85
10	5946	50	700	≥ 0.85

- Dominating term of success is $qe^{-\delta s}$ for $q \leq 100$.
- If victim sends ≈ 50 messages per key then for $\delta = 8, s = 1485$ we can recover $600 > 576$ key bits with probability ≥ 0.85.

Success of Our Attack

δ	s	q	k	Pr
2	21	50	300	≥ 0.88
4	91	50	400	≥ 0.85
6	369	50	500	≥ 0.85
8	1485	50	600	≥ 0.85
10	5946	50	700	≥ 0.85

- Dominating term of success is $qe^{-\delta s}$ for $q \leq 100$.
- If victim sends ≈ 50 messages per key then for $\delta = 8, s = 1485$ we can recover $600 > 576$ key bits with probability ≥ 0.85.
- $s = 1485$ too large (increased energy $\rightarrow$ OS detection)

Success of Our Attack

δ	s	q	k	Pr
2	21	50	300	≥ 0.88
4	91	50	400	≥ 0.85
6	369	50	500	≥ 0.85
8	1485	50	600	≥ 0.85
10	5946	50	700	≥ 0.85

- Dominating term of success is $qe^{-\delta s}$ for $q \leq 100$.
- If victim sends ≈ 50 messages per key then for $\delta = 8, s = 1485$ we can recover $600 > 576$ key bits with probability ≥ 0.85.
- $s = 1485$ too large (increased energy $\rightarrow$ OS detection)
- More modest approach: 500 key bits using $\delta = 6, s = 369$ with probability ≥ 0.85 (computationally cheaper).

Implications of Our Attack

- Telegram's claim - code builds are reproducible $\rightarrow$ difficult to massively roll out our attack.

Implications of Our Attack

- Telegram's claim - code builds are reproducible $\rightarrow$ difficult to massively roll out our attack.
- The attack can still be deployed on targeted users or closed-source third party clients.

Conclusions

- We introduced the first algorithm substitution attack on MTProto 2.0:

Conclusions

- We introduced the first algorithm substitution attack on MTProto 2.0:
  - We showed that the attack is undetectable.

Conclusions

- We introduced the first algorithm substitution attack on MTProto 2.0:
  - We showed that the attack is undetectable.
  - We showed that for $q \geq |K|/(4+\delta)$, we can extract 10 key bits per message with probability at least 0.85 - enough to recover most of the key bits for the encryption pass.

Conclusions

- We introduced the first algorithm substitution attack on MTProto 2.0:
  - We showed that the attack is undetectable.
  - We showed that for $q \geq |K|/(4+\delta)$, we can extract 10 key bits per message with probability at least 0.85 - enough to recover most of the key bits for the encryption pass.
- In the full paper, we show that the subversion attack can be averted (modified version):

Conclusions

- We introduced the first algorithm substitution attack on MTProto 2.0:
  - We showed that the attack is undetectable.
  - We showed that for $q \geq |K|/(4+\delta)$, we can extract 10 key bits per message with probability at least 0.85 - enough to recover most of the key bits for the encryption pass.
- In the full paper, we show that the subversion attack can be averted (modified version):
  - MTProto 2.0 can be seen as an instantiation of a secure DAE scheme (MTProto-G).

Conclusions

- We introduced the first algorithm substitution attack on MTProto 2.0:
  - We showed that the attack is undetectable.
  - We showed that for $q \geq |K|/(4+\delta)$, we can extract 10 key bits per message with probability at least 0.85 - enough to recover most of the key bits for the encryption pass.
- In the full paper, we show that the subversion attack can be averted (modified version):
  - MTProto 2.0 can be seen as an instantiation of a secure DAE scheme (MTProto-G).
  - Small changes to algorithm (mainly padding) $\rightarrow$ deterministic.

Conclusions

- We introduced the first algorithm substitution attack on MTProto 2.0:
  - We showed that the attack is undetectable.
  - We showed that for $q \geq |K|/(4+\delta)$, we can extract 10 key bits per message with probability at least 0.85 - enough to recover most of the key bits for the encryption pass.
- In the full paper, we show that the subversion attack can be averted (modified version):
  - MTProto 2.0 can be seen as an instantiation of a secure DAE scheme (MTProto-G).
  - Small changes to algorithm (mainly padding) $\rightarrow$ deterministic.
  - Assuming perfect decryptability & key-independent messages $\rightarrow$ modified version is subversion-resistant.

Thank You!

References I

[Alb+22] Martin R. Albrecht et al. "Four Attacks and a Proof for Telegram". In: *Security and Privacy – IEEE-S&P 2022, Proceedings*. 2022, pp. 87–106.

[AP19] Marcel Armour and Bertram Poettering. "Subverting Decryption in AEAD". In: *Cryptography and Coding – IMACC 2019, Proceedings*. 2019, pp. 22–41.

[BJK15] Mihir Bellare, Joseph Jaeger, and Daniel Kane. "Mass-Surveillance without the State: Strongly Undetectable Algorithm-Substitution Attacks". In: *Computer and Communications Security – ACM-CCS 2015, Proceedings*. 2015, 1431–1440.

[BPR14] Mihir Bellare, Kenneth G. Paterson, and Phillip Rogaway. "Security of Symmetric Encryption against Mass Surveillance". In: *Advances in Cryptology – CRYPTO 2014, Proceedings*. 2014, pp. 1–19.

References II

[JO16] Jakob Jakobsen and Claudio Orlandi. "On the CCA (in)Security of MTProto". In: *Security and Privacy in Smartphones and Mobile Devices – SPSM@CCS 2016, Proceedings*. 2016, pp. 113–116.

[YY96] Adam L. Young and Moti Yung. "The Dark Side of "Black-Box" Cryptography, or: Should We Trust Capstone?" In: *Advances in Cryptology – CRYPTO 1996, Proceedings*. 1996, pp. 89–103.

[YY97] Adam L. Young and Moti Yung. "Kleptography: Using Cryptography Against Cryptography". In: *Advances in Cryptology – EUROCRYPT 1997, Proceeding*. 1997, pp. 62–74.