NTNU | Norwegian University of Science and Technology

PROTOCOL COMPOSITION 3

TTM4205 – Lecture 17

Tjerand Silde

07.11.2023

Contents

Contents

Project Presentations

Most of you will be presenting on November 23rd, but the program is a bit full with 11 groups total. Send me an email if you want or have to present on November 21st instead.

Reference Group

The minutes from last reference group meeting is available on the wiki. We will have a last meeting in a few weeks.

Course Evaluation

The department sent you a course evaluation on email.
Please answer the questionnaire, it is very valuable to us.

The Remaining Schedule

45	7/11	Lecture	Tjerand	Protocol Composition 3	Slides
45	7/11	Lab/Ex	Jonathan	Assignments	
45	9/11	Lecture	Tjerand	Course Summary	
46	14/11	Lecture	Tjerand	Guest Lecture: Håkon Jacobsen	
46	14/11	Lab/Ex	Tjerand	Assignments	
46	16/11	Lecture	Tjerand	Guest Lecture: Oskar Goldhahn	
47	21/11	Lab/Ex	Jonathan	Assigments	
47	21/11	Lecture	Tjerand	Guest Lecture: Vadim Lyubashevsky	
47	23/11	Lecture	Tjerand	***Project Presentations***	

Contents

Quick recall of ElGamal

Encryption scheme: variant of ElGamal in $\mathbb{Z}/p\mathbb{Z}$, with safe primes (Sophie Germain).

Let g be a generator, and $(\mathrm{sk}, \mathrm{pk} = g^{\mathrm{sk}})$ a key-pair.

The **plain ElGamal encryption** of a message m is:

$$\mathsf{Enc}_{g,\mathrm{pk}}(m) = (a, b) = (g^r, \mathrm{pk}^r \cdot m)\,,$$

where r is a random (to be used only once).

The **decryption** using sk is:

$$\mathsf{Dec}_{g,\mathrm{sk}}(a, b) = b \cdot a^{-\mathrm{sk}} = m.$$

If done correctly, this gives IND-CPA security.

A triple-ElGamal (encryption)

In the original scheme, there is a **multi-level** variant:
Choose $p_1 < p_2 < p_3$, three safe primes, together with 3 generators g_1, g_2, g_3.

The **keys** are

$$\mathrm{sk} = (\mathrm{sk}_1, \mathrm{sk}_2, \mathrm{sk}_3); \quad \mathrm{pk} = (g_1^{\mathrm{sk}_1}, g_2^{\mathrm{sk}_2}, g_3^{\mathrm{sk}_3}),$$

The **encryption** of a message $m \in \mathbb{Z}/p_1\mathbb{Z}$ is obtained by

$$\begin{aligned} (a_1, b_1) &:= \mathsf{Enc}_{g_1,\mathrm{pk}_1}(m); \quad \text{map } a_1 \text{ to } \mathbb{Z}/p_2\mathbb{Z}; \\ (a_2, b_2) &:= \mathsf{Enc}_{g_2,\mathrm{pk}_2}(a_1); \quad \text{map } a_2 \text{ to } \mathbb{Z}/p_3\mathbb{Z}; \\ (a_3, b_3) &:= \mathsf{Enc}_{g_3,\mathrm{pk}_3}(a_2), \end{aligned}$$

and the encrypted message is

$$\mathsf{MultiEnc}(m) = (b_1, b_2, a_3, b_3).$$

Rem. All mapping are obtained by canonical lifting to $\mathbb{Z}$.

A triple-ElGamal (decryption)

Knowing sk, the operations can be reversed to **decrypt** m from (b_1, b_2, a_3, b_3):

$$\begin{aligned} a_2 &:= \mathrm{Dec}_{g_3,\mathrm{sk}_3}(a_3, b_3); \quad \text{map } a_2 \text{ to } \mathbb{Z}/p_2\mathbb{Z}; \\ a_1 &:= \mathrm{Dec}_{g_2,\mathrm{sk}_2}(a_2, b_2); \quad \text{map } a_1 \text{ to } \mathbb{Z}/p_1\mathbb{Z}; \\ m &:= \mathrm{Dec}_{g_1,\mathrm{sk}_1}(a_1, b_1). \end{aligned}$$

Due to the inequality $p_1 < p_2 < p_3$, **this works**.

A triple-ElGamal (decryption)

Knowing sk, the operations can be reversed to **decrypt** m from (b_1, b_2, a_3, b_3):

$$\begin{aligned} a_2 &:= \mathrm{Dec}_{g_3,\mathrm{sk}_3}(a_3, b_3); \quad \text{map } a_2 \text{ to } \mathbb{Z}/p_2\mathbb{Z}; \\ a_1 &:= \mathrm{Dec}_{g_2,\mathrm{sk}_2}(a_2, b_2); \quad \text{map } a_1 \text{ to } \mathbb{Z}/p_1\mathbb{Z}; \\ m &:= \mathrm{Dec}_{g_1,\mathrm{sk}_1}(a_1, b_1). \end{aligned}$$

Due to the inequality $p_1 < p_2 < p_3$, **this works**.

Security.
Contrary to triple-DES where the number of operations to break the system is squared, here it is just multiplied by 3.

Breaking the scheme is not harder than to

break the 3 underlying ElGamal independently.

DLP with CADO-NFS

Running times on my 4-year old nothing-special desk PC:

key number	time
sk_1	425 sec
sk_2	507 sec
sk_3	314 sec

Each line includes 2 runs of CADO-NFS (one for g_i, one for pk_i); but many steps are (automatically) shared.

Figure: They used 256-bit finite field ElGamal...
https://rwc.iacr.org/2020/slides/Gaudry.pdf

Contents

Three Lessons From Threema: Analysis of a Secure Messenger

Kenneth G. Paterson
Applied Cryptography Group, ETH Zurich

Matteo Scarlata
Applied Cryptography Group, ETH Zurich

Kien Tuong Truong
Applied Cryptography Group, ETH Zurich

Figure: `https://breakingthe3ma.app/files/Threema-PST22.pdf`

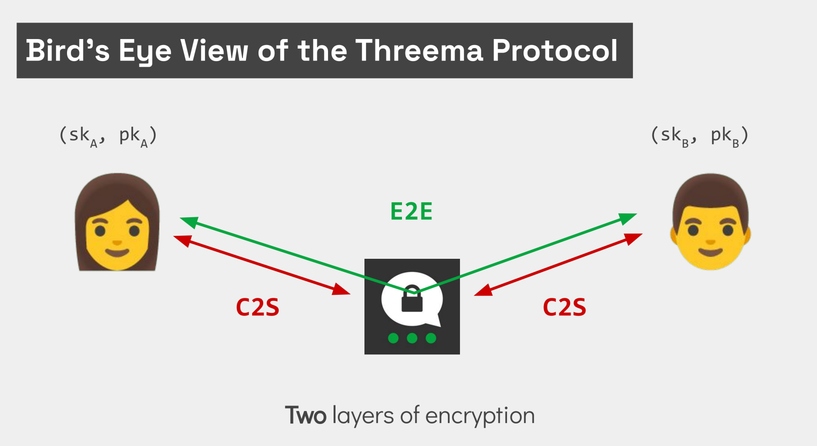
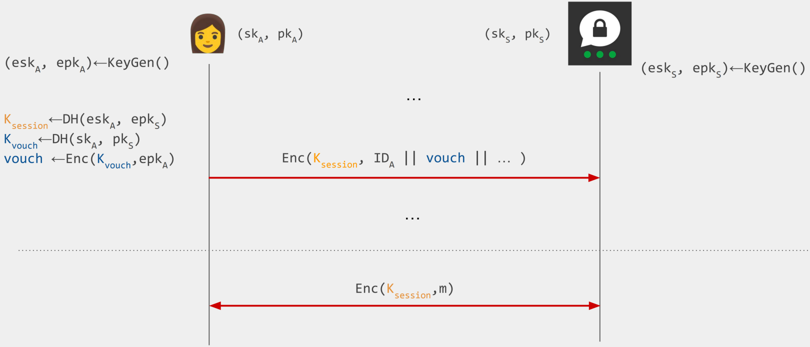


Figure: https://iacr.org/submit/files/slides/2023/rwc/rwc2023/75/slides.pdf

The C2S Protocol*



* Simplified, details omitted

The C2S Protocol: Vouch Box

$K_{vouch} \leftarrow DH(sk_A,\ pk_S)$ DH(long-term, long-term)

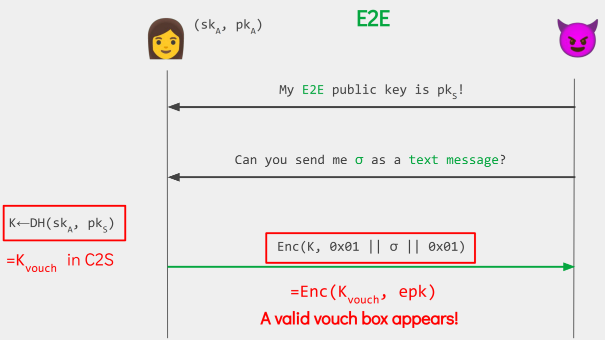
$vouch \leftarrow Enc(K_{vouch}, epk_A)$ Enc(some value)

What if we could find a special keypair (esk, epk) such that:

epk = 0x01 || σ || 0x01

UTF-8 valid string of 30B

Attacking the C2S Protocol



Part 1: Getting That Key

```
epk = 0x01 || σ || 0x01
```

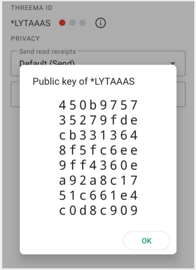
UTF-8 valid string of 30B

Requires sampling 2^{51} keys!

Part 2: The Bamboozling

- Threema Gateway: paid API
- Can register accounts **with arbitrary public keys**
- **Without proof of possession** of the corresponding private key!

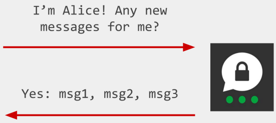
=> ***LYTAAAS**




```
public static final byte[] SERVER_PUBKEY = new byte[] {
    (byte) 0x45, (byte) 0x0b, (byte) 0x97, (byte) 0x57,
    (byte) 0x35, (byte) 0x27, (byte) 0x9f, (byte) 0xde,
    (byte) 0xcb, (byte) 0x33, (byte) 0x13, (byte) 0x64,
    (byte) 0x8f, (byte) 0x5f, (byte) 0xc6, (byte) 0xee,
    (byte) 0x9f, (byte) 0xf4, (byte) 0x36, (byte) 0x0e,
    (byte) 0xa9, (byte) 0x2a, (byte) 0x8c, (byte) 0x17,
    (byte) 0x51, (byte) 0xc6, (byte) 0x61, (byte) 0xe4,
    (byte) 0xc0, (byte) 0xd8, (byte) 0xc9, (byte) 0x09
};
```

Vouch Box Forgery

- **C2S x E2E** cross-protocol attack:
- Sending a text message... compromises client authentication **forever**!



Attack: Vouch Box Forgery

Contents

Four Attacks and a Proof for Telegram*

Martin R. Albrecht[1], Lenka Mareková[2], Kenneth G. Paterson[3], and Igors Stepanovs[3]

[1] King's College London
martin.albrecht@kcl.ac.uk
[2] Information Security Group, Royal Holloway, University of London
lenka.marekova.2018@rhul.ac.uk
[3] Applied Cryptography Group, ETH Zurich
{kenny.paterson,istepanovs}@inf.ethz.ch

31 March 2023

Figure: https://eprint.iacr.org/2023/469.pdf

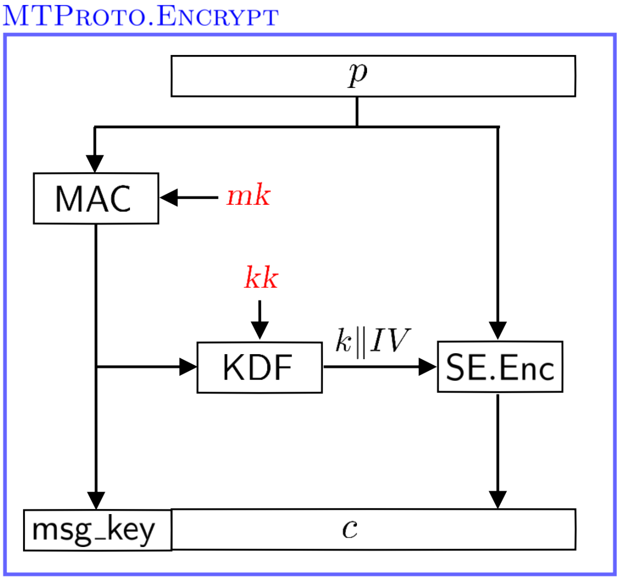
MTProto

The **MTProto** protocol is not well-studied:

- **2013: Telegram launched** with MTProto 1.0.
- **2016: Jakobsen** and **Orlandi** showed that MTProto 1.0 is not CCA-secure.
- **2017: Telegram released MTProto 2.0** that addressed the security concerns.
- **2017: Sušánka** and **Kokeš** reported an attack based on improper validation in the Android client.
- **2018: Kobeissi** reported input validation bugs in Telegram's Windows Phone client.
- **2020: Miculan** and **Vitacolonna** proved MTProto 2.0 secure in a symbolic model, assuming ideal building blocks.

Figure: `https://iacr.org/submit/files/slides/2022/rwc/rwc2022/60/slides.pdf`

MTProtoEncrypt



MTProtoEncrypt

supplied by attacker

If (msg_length > length) then … // **Android**

Outcome of comparison depends on 32 bits on msg_length.

If comparison fails: two conditional jumps added.

If (msg_length > 2^{24}) then … // **Desktop**

Outcome of comparison depends on 8 bits on msg_length.

If comparison fails: MAC verification is omitted.

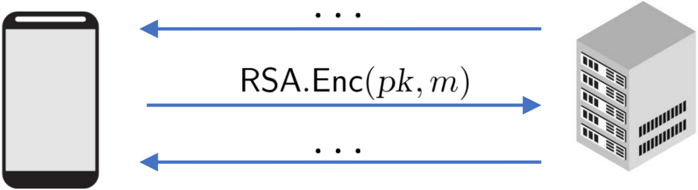
If not ($12 \leq \ell -$ msg_length ≤ 1024) then … // **iOS**

Outcome of comparison depends on 32 bits on msg_length.

If comparison fails: MAC verification takes a shorter input.

MTProtoEncrypt

We attack **Telegram**'s key exchange.



Telegram uses textbook **RSA** encryption.

$$m := \mathsf{SHA\text{-}1}(\mathsf{data}) \| \mathsf{data} \| \mathsf{padding}$$

Four Attacks

- Message reordering (lack of metadata authentication)
- Re-encryption of dropped messages lead to CPA attacks
- Timing attack against encrypt and mac using AES-IGE
- RSA padding oracle using textbook RSA with SHA-1

Future Work

Large parts of **Telegram**'s design remain unstudied:

Secret chats (including encrypted voice and video calls).
The key exchange.

Multi-user security.
Forward secrecy.

Telegram Passport.
Bot APIs.

The higher-level message processing.
Control messages.

Encrypted CDNs.
Cloud storage.

These are pressing topics for future work.

Contents

Mesh Messaging in Large-scale Protests: Breaking Bridgefy

Martin R. Albrecht, Jorge Blasco, Rikke Bjerg Jensen, and Lenka Mareková

Royal Holloway, University of London
{martin.albrecht,jorge.blascoalis,rikke.jensen,lenka.marekova}@rhul.ac.uk

Figure: https://eprint.iacr.org/2021/214.pdf

Bridgefy (Again)

Breaking Bridgefy, again:
Adopting libsignal is not enough

Martin R. Albrecht
Information Security Group,
Royal Holloway, University of London

Raphael Eikenberg
Applied Cryptography Group,
ETH Zurich

Kenneth G. Paterson
Applied Cryptography Group,
ETH Zurich

Figure:
https://www.usenix.org/system/files/sec22fall_albrecht.pdf

Jitsi

Practically-exploitable Vulnerabilities in the Jitsi Video Conferencing System

Robertas Maleckas
ETH Zürich
Switzerland
robertas.maleckas@alumni.ethz.ch

Kenneth G. Paterson
ETH Zürich
Switzerland
kenny.paterson@inf.ethz.ch

Martin R. Albrecht
King's College London
UK
martin.albrecht@kcl.ac.uk

Figure: https://eprint.iacr.org/2023/1118.pdf

Matrix

Practically-exploitable Cryptographic Vulnerabilities in Matrix

Martin R. Albrecht*, Sofía Celi†, Benjamin Dowling‡ and Daniel Jones§
* King's College London, martin.albrecht@kcl.ac.uk
† Brave Software, cherenkov@riseup.net
‡ Security of Advanced Systems Group, University of Sheffield, b.dowling@sheffield.ac.uk
§Information Security Group, Royal Holloway, University of London, dan.jones@rhul.ac.uk

Figure:
https://nebuchadnezzar-megolm.github.io/static/paper.pdf

Contents

Conclusions

Conclusions

- End-to-end security is hard

Conclusions

- End-to-end security is hard
- Composing protocols is hard

Conclusions

- End-to-end security is hard
- Composing protocols is hard
- Have very clear descriptions

Conclusions

- End-to-end security is hard
- Composing protocols is hard
- Have very clear descriptions
- Always (try to) prove security

Conclusions

- End-to-end security is hard
- Composing protocols is hard
- Have very clear descriptions
- Always (try to) prove security
- Use up-to-date modern primitives

Conclusions

- End-to-end security is hard
- Composing protocols is hard
- Have very clear descriptions
- Always (try to) prove security
- Use up-to-date modern primitives
- Be careful about reusing primitives

Conclusions

- End-to-end security is hard
- Composing protocols is hard
- Have very clear descriptions
- Always (try to) prove security
- Use up-to-date modern primitives
- Be careful about reusing primitives
- Authenticate all messages and metadata

Conclusions

- End-to-end security is hard
- Composing protocols is hard
- Have very clear descriptions
- Always (try to) prove security
- Use up-to-date modern primitives
- Be careful about reusing primitives
- Authenticate all messages and metadata
- Always use ephemeral keys for sessions

Conclusions

The Signal Protocol and TLS 1.3 are two out of few protocols that we got right. It took many years of research, analysis, attacks and experience to get it right in the end.

Questions?