Cryptography in the Wild

Kenny Paterson
Applied Cryptography Group
ETH Zurich

ViSP
June 28 2023

Overview

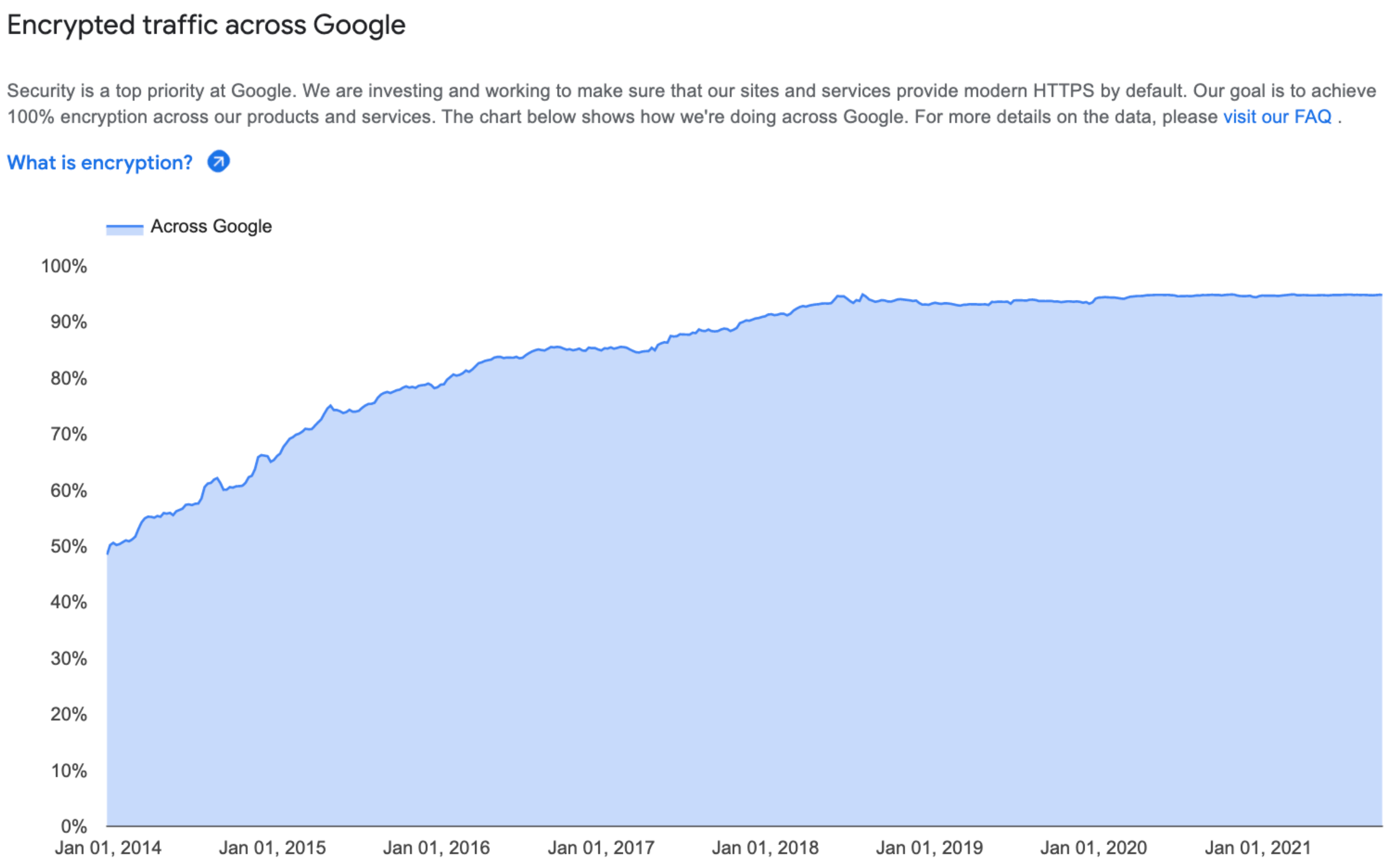
APPLIED CRYPTO GROUP

- The spread of cryptography in the wild
- The what
- The how
- Two case studies – Threema and MEGA
- The challenges
- The why
- Concluding remarks

The Spread of Cryptography in the Wild

- e-commerce
- social media
- online personal banking
- payment systems
- secure messaging (WhatsApp, Signal, Telegram, Threema,…)
- mobile telephony
- gov/mil comms systems
- video conferencing
- SSH
- VPN/remote access
- disk encryption
- cloud storage
- TPMs, SGX, …
- identity-card systems
- IDM systems
- ticketing systems
- DRM
- cryptocurrencies
- crypto-custody solutions
- PrivacyPass
- Prio - privacy-preserving browser analytics.
- privacy-preserving contact tracing (GAEN, DP3T)
- …

The Spread of Cryptography in the Wild



Encrypted traffic across Google

Security is a top priority at Google. We are investing and working to make sure that our sites and services provide modern HTTPS by default. Our goal is to achieve 100% encryption across our products and services. The chart below shows how we're doing across Google. For more details on the data, please visit our FAQ .

What is encryption?

https://transparencyreport.google.com/https/overview?hl=en

Crypto in the Wild – The What

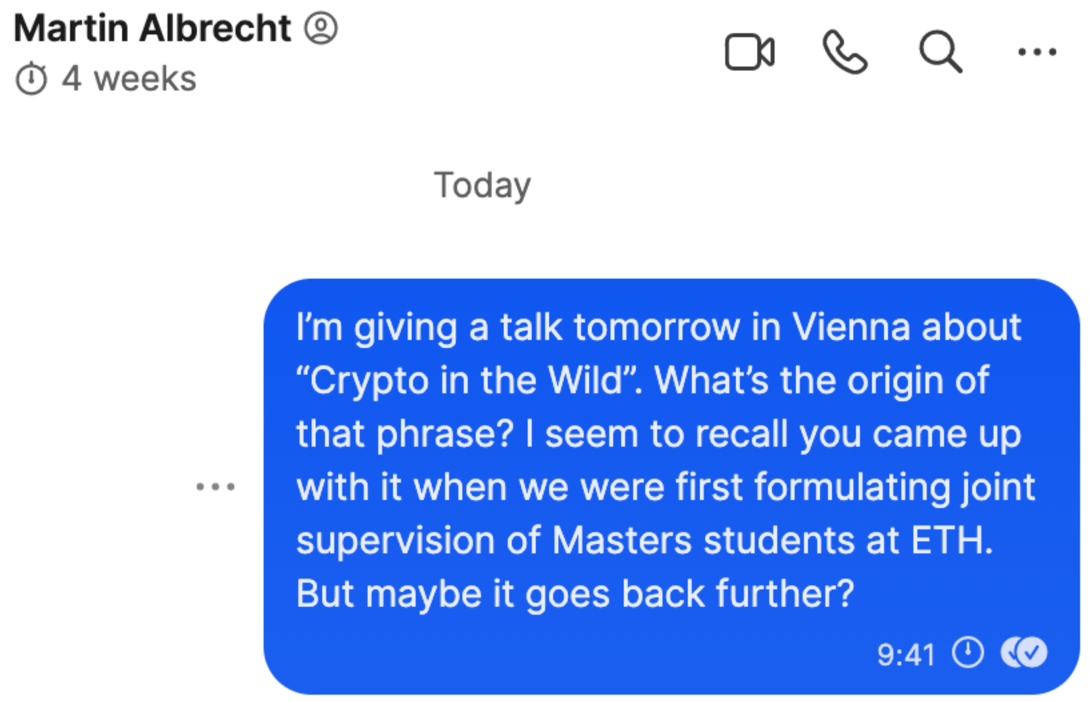
APPLIED CRYPTO GROUP

- Find interesting examples of cryptography being used in standards, products, deployed systems.
- Analyse them to:
  - Find and responsibly disclose vulnerabilities

AND/OR

  - Build security models and proofs for deployed systems.
  - Sometimes identify new cryptographic problems for further research.
  - Write papers, build relationships with vendors, train students, improve the quality of deployed cryptography.

Origins of “Crypto in the Wild”



Martin Albrecht

4 weeks

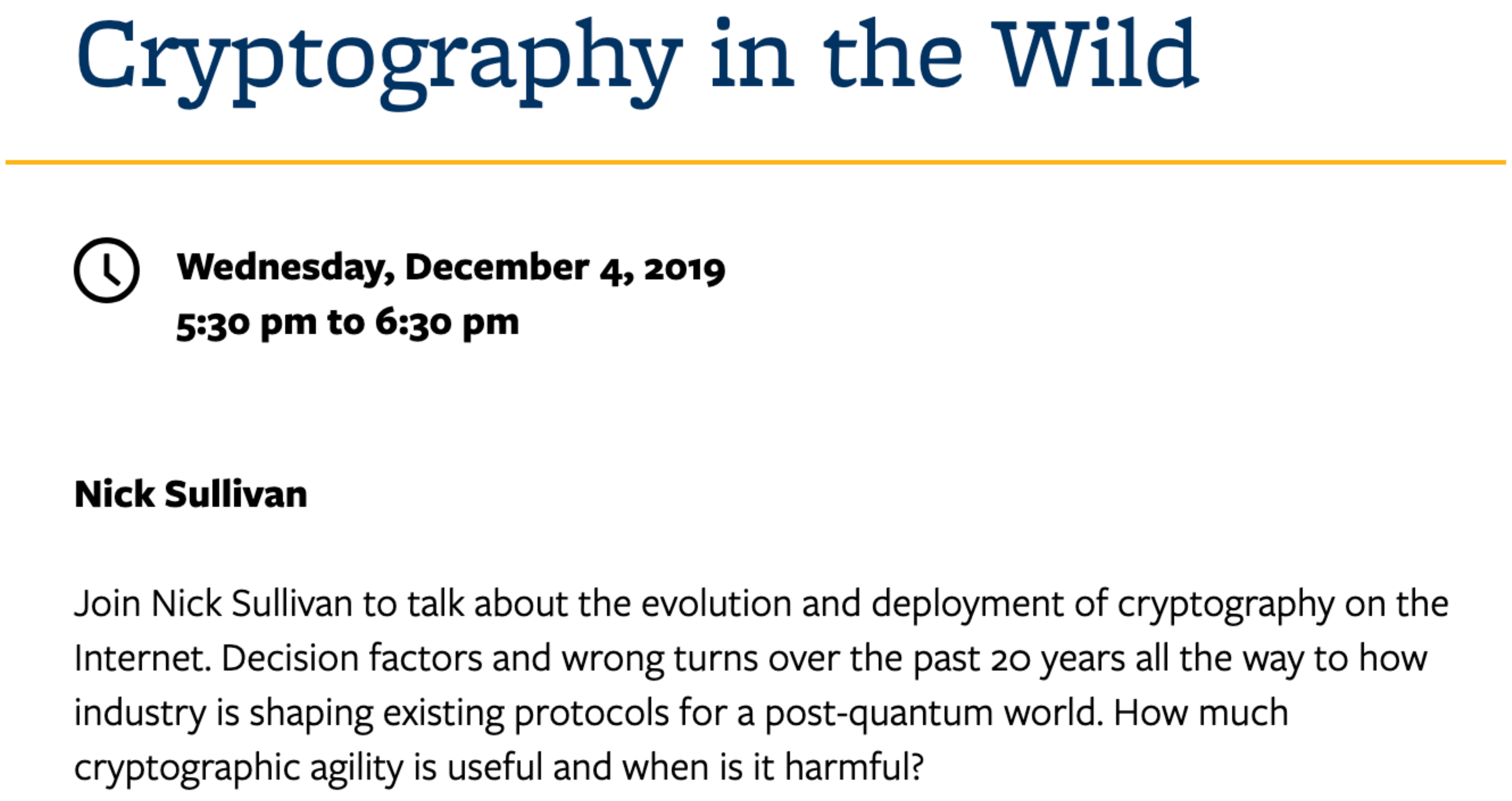
Today

I’m giving a talk tomorrow in Vienna about “Crypto in the Wild”. What’s the origin of that phrase? I seem to recall you came up with it when we were first formulating joint supervision of Masters students at ETH. But maybe it goes back further?

9:41

Origins of “Crypto in the Wild”

APPLIED CRYPTO GROUP



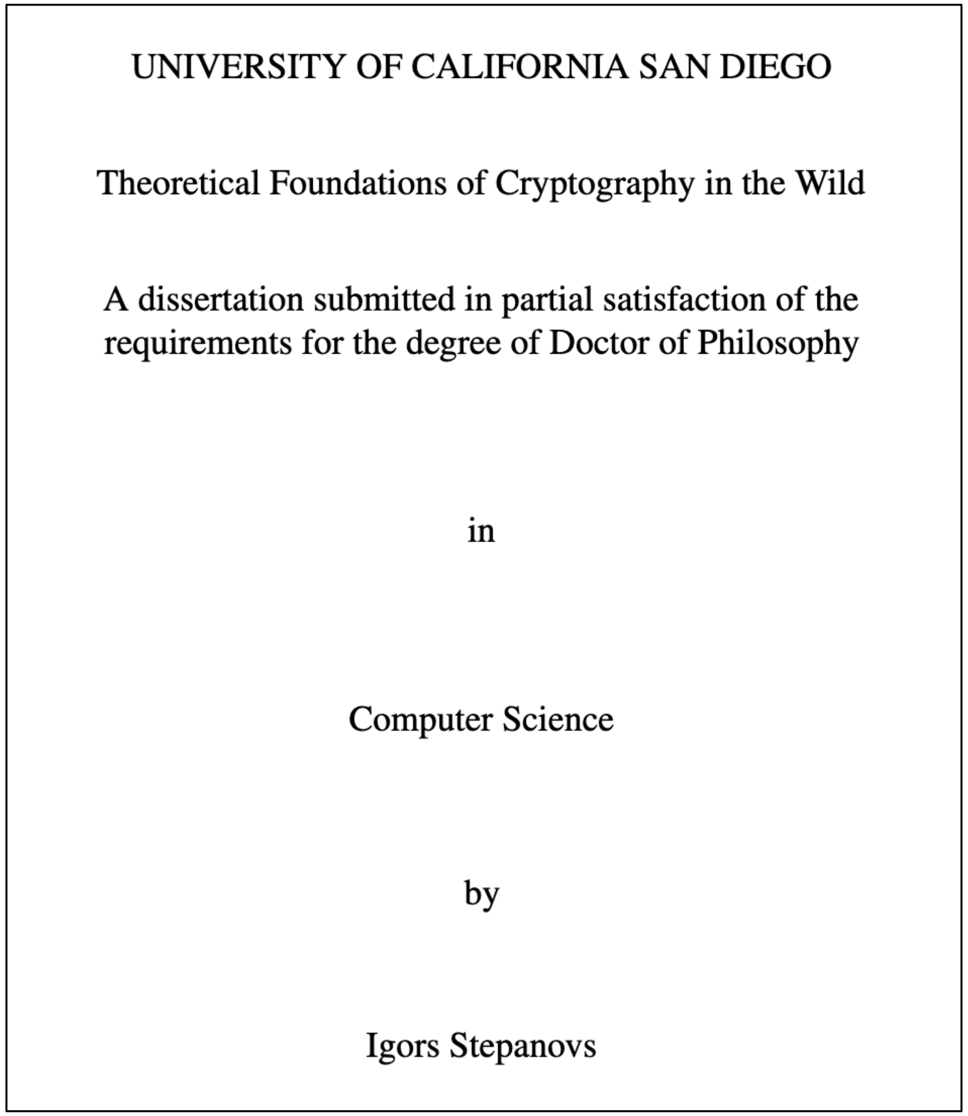
Cryptography in the Wild

Wednesday, December 4, 2019
5:30 pm to 6:30 pm

Nick Sullivan

Join Nick Sullivan to talk about the evolution and deployment of cryptography on the Internet. Decision factors and wrong turns over the past 20 years all the way to how industry is shaping existing protocols for a post-quantum world. How much cryptographic agility is useful and when is it harmful?

Origins of “Crypto in the Wild”


UNIVERSITY OF CALIFORNIA SAN DIEGO

Theoretical Foundations of Cryptography in the Wild

A dissertation submitted in partial satisfaction of the requirements for the degree of Doctor of Philosophy

in

Computer Science

by

Igors Stepanovs

https://www.igors.org/

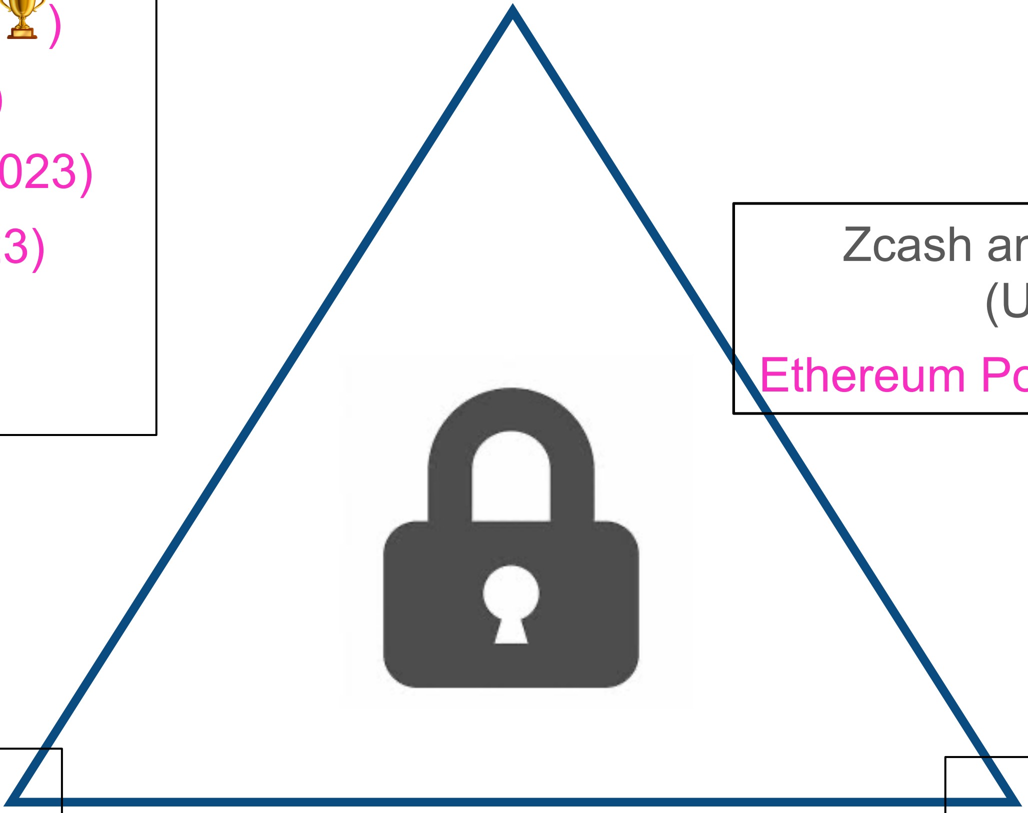
Recent Crypto in the Wild Projects

APPLIED CRYPTO GROUP

Telegram (IEEE S&P 2022 🏆)
Bridgefy – again (USENIX Sec 2022)
OpenPGP (ACM CCS 2022 🏆)
EDHOC (Euro S&P 2023)
Telegram – again (AsiaCCS 2023)
Threema (USENIX Sec 2023)
Jitsi Meet (disclosed)
Briar (disclosed)

Data in transit
aka secure communications

Pink indicates a Master's student project at ETH



Zcash and Monero side-channels (USENIX Sec 2020)
Ethereum PoW sync (USENIX Sec 2023)

Data at rest
aka secure storage

MEGA (IEEEE S&P 2023 🏆)
MEGA – again (EuroCrypt 2023)
NextCloud (disclosed)

Data under computation

MongoDB Queryable Encryption (USENIX Sec 2023)

Crypto in the Wild – The How

Criteria for target selection:

- Use of non-trivial and/or non-standard cryptography
- Size of user base
- Local importance
- Availability of source code
- Legality of reverse engineering
- Availability of a whitepaper describing crypto architecture
- Strength/credibility of security claims made by vendor
- “Smell”

Crypto in the Wild – The How

Process (attack mode):

- Define target and objectives
- Develop relevant threat model(s)
- Read the whitepaper/code
- **Do:**
  - Build pseudo-code models of the cryptographic “core”
  - Write down attack ideas on paper
  - Build and test Proof-of-Concepts (PoCs) for the attacks
- **Until** enough attacks have been found
- Write a research paper, run disclosure process

Crypto in the Wild – The How

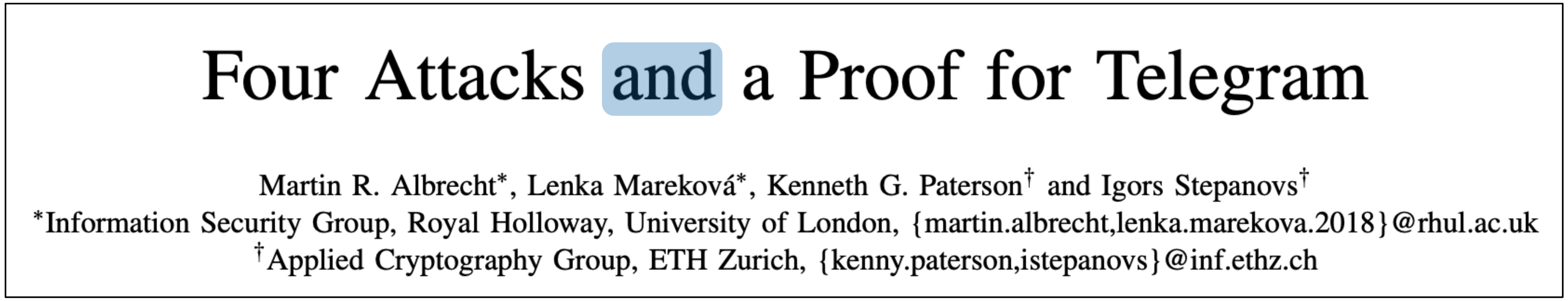
Process (proof mode):

- Define target and objectives
- Develop relevant threat model(s)
- Read the whitepaper/code
- **Do:**
  - Build pseudo-code models of the cryptographic “core”
  - Develop suitable security model(s)
  - Write security proofs
- **Until** enough proofs have been produced
- Write a research paper

Crypto in the Wild – The How

$$\left|\begin{array}{l}\text{attack}\\ \text{mode}\end{array}\right\rangle + \left|\begin{array}{l}\text{proof}\\ \text{mode}\end{array}\right\rangle$$

Crypto in the Wild – The How



Four Attacks and a Proof for Telegram

Martin R. Albrecht*, Lenka Mareková*, Kenneth G. Paterson† and Igors Stepanovs†
*Information Security Group, Royal Holloway, University of London, {martin.albrecht,lenka.marekova.2018}@rhul.ac.uk
†Applied Cryptography Group, ETH Zurich, {kenny.paterson,istepanovs}@inf.ethz.ch

IEEE Security and Privacy Symposium 2022

Distinguished paper award

https://mtpsym.github.io/

Crypto in the Wild – The How

Some things to look out for when in attack mode:

- ECB mode
- Exotic + home-made encryption modes
- Lack of integrity mechanisms
- Improper use of integrity, e.g. MtE, E&M
- Padding oracle attacks
- Nonce reuse
- Lack of proper key separation/key reuse problems
- Bad interactions between different protocols
- Bespoke RSA padding schemes
- Roll-your-own authentication and key exchange protocols
- Naïve use of NaCl and other libraries
- Use of weak PRNGs or homebrew randomness generation methods
- Compression combined with encryption
- ….

Crypto in the Wild – Examples

Brief coverage of a couple of case studies.

- Threema – a “Swiss-made” secure messenger.
- MEGA – an E2EE cloud storage provider.

What is Threema?

- An “end-to-end encrypted instant messaging application” for Android and iOS
- Released in 2012

*“Threema is **100% Swiss Made**, hosts its own servers in Switzerland, and, **unlike US services** (which are subject to the CLOUD Act, for example), **it is fully GDPR-compliant.**"*

Who Uses Threema?

- 11 million private users worldwide [1]
- Various organizations and political entities:

Schweizerische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confederaziun svizra

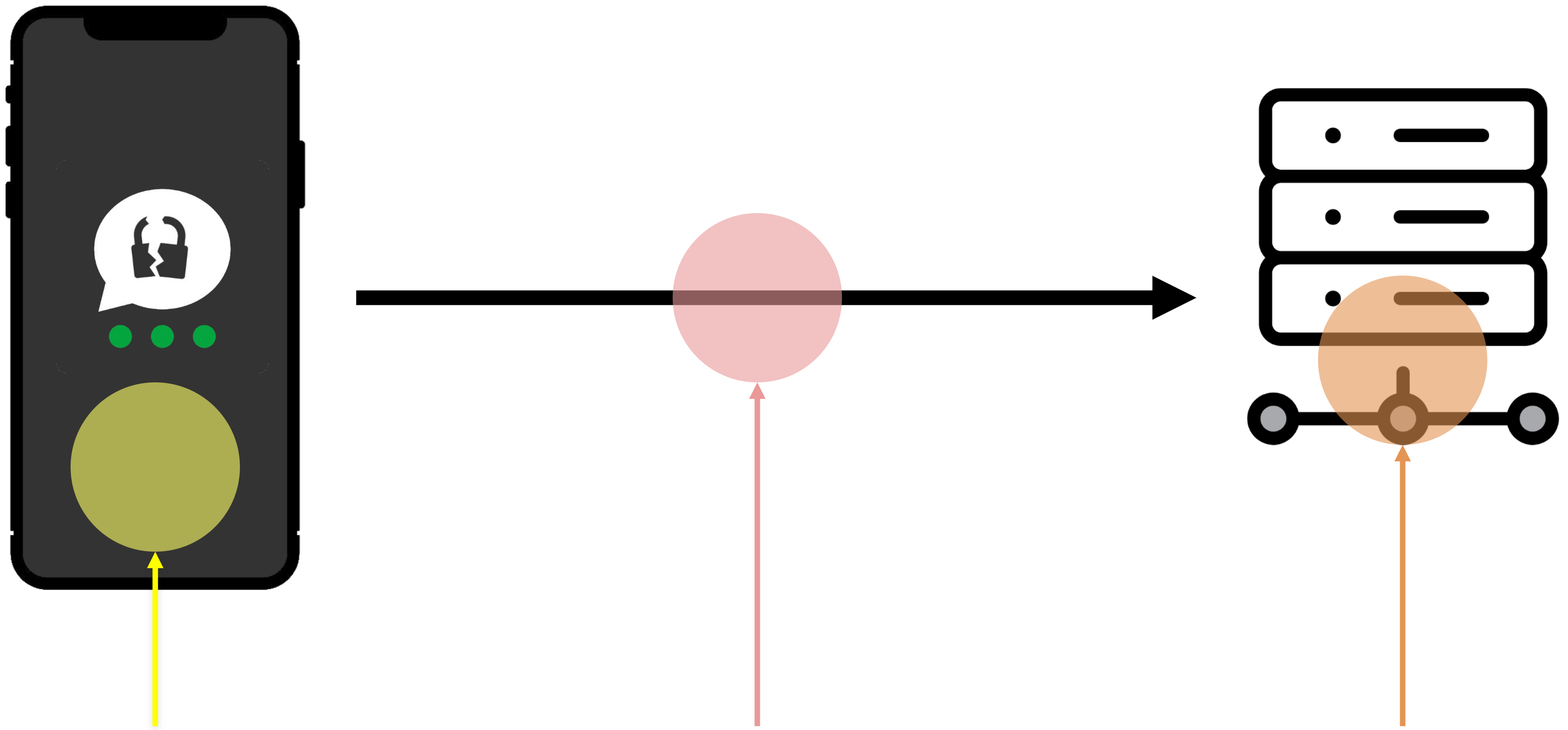
Mercedes-Benz

DEUTSCHE BUNDESBANK EUROSYSTEM

Emirates

[1] https://threema.ch/en/about (Last checked 19 Mar 2023)

Threat Models

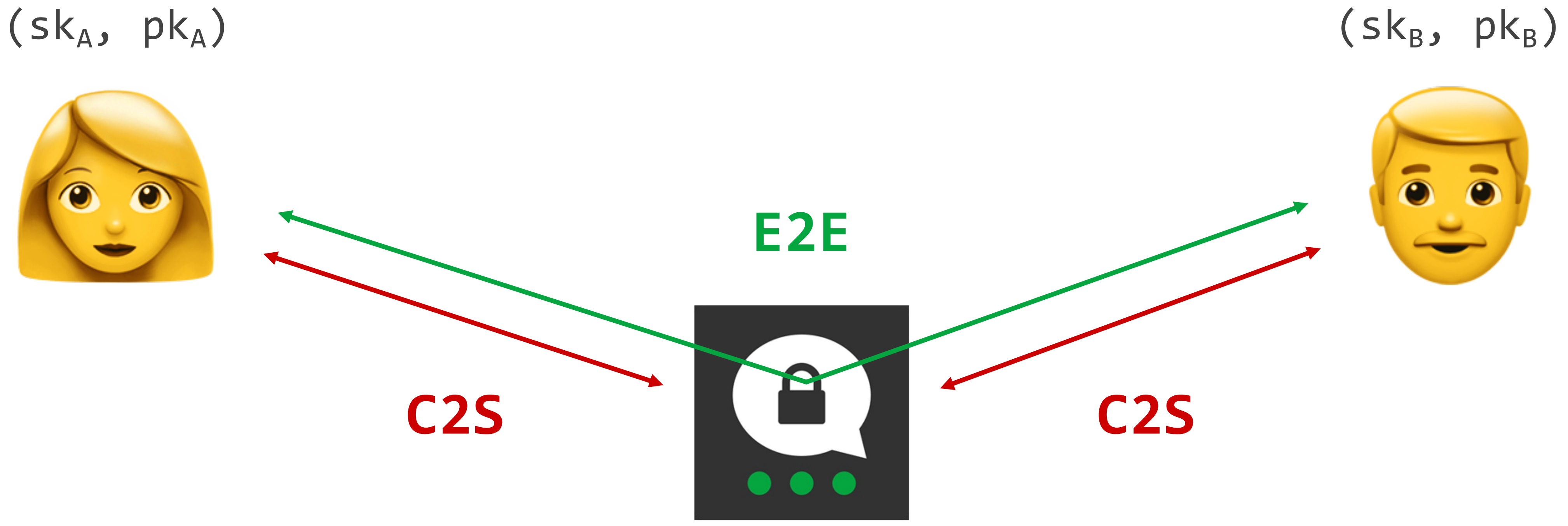


Compelled Access

External Actor

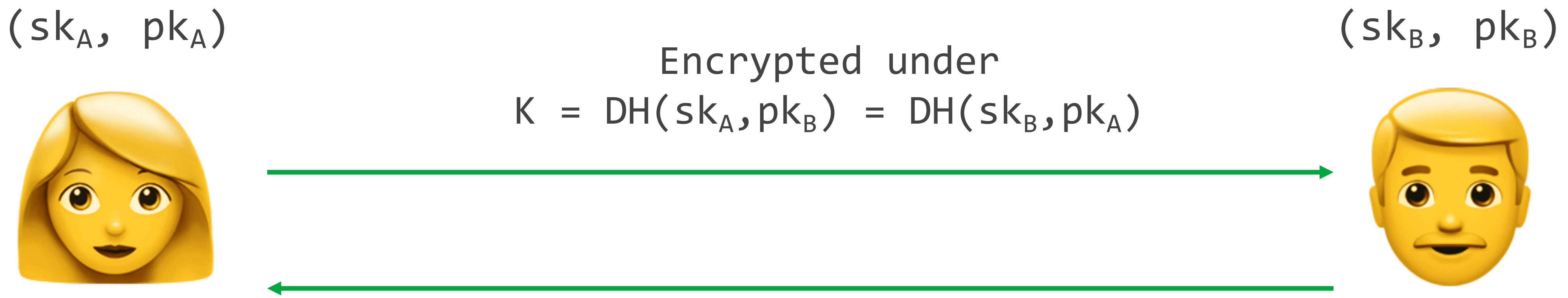
Compromised Threema server

High-level View of Threema



Two layers of encryption

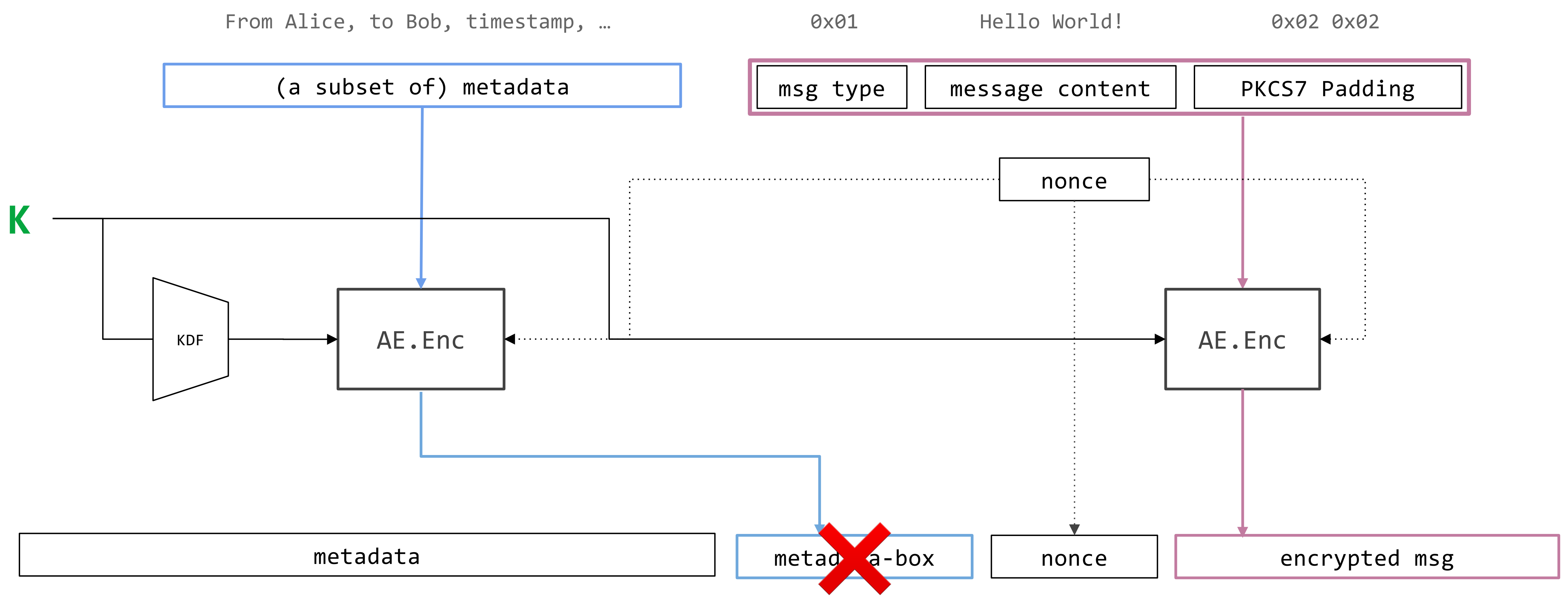
E2E Protocol



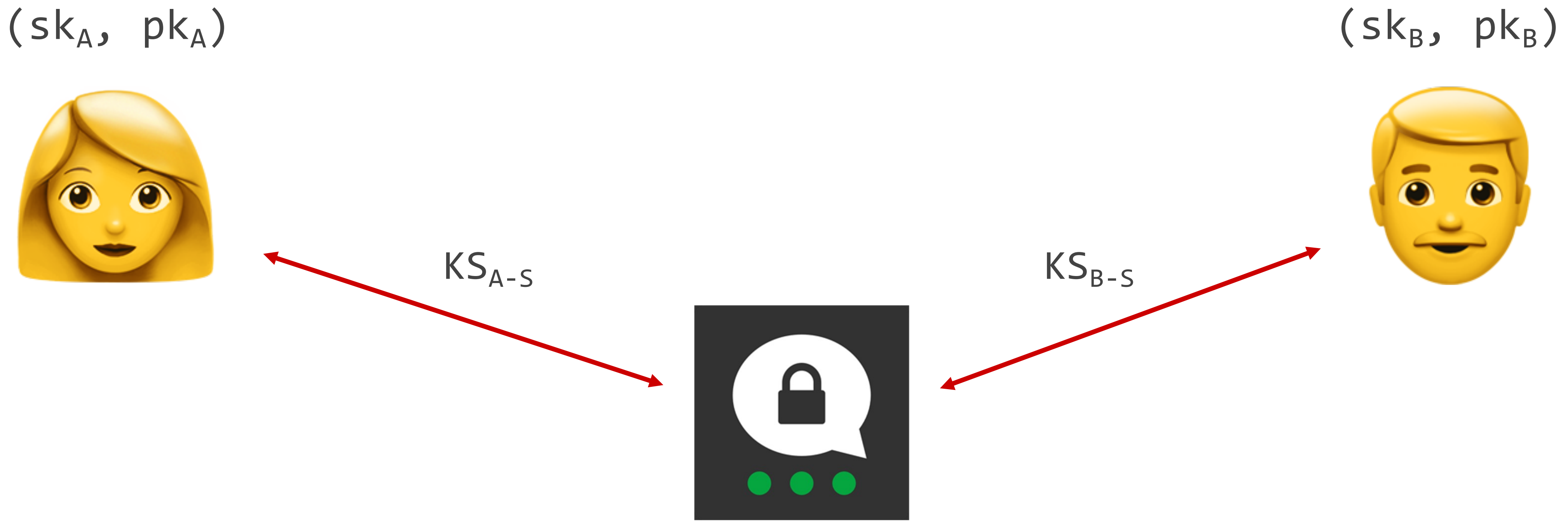
No Forward Secrecy! ❌

No Post-Compromise Security! ❌

E2E Protocol: Message Format



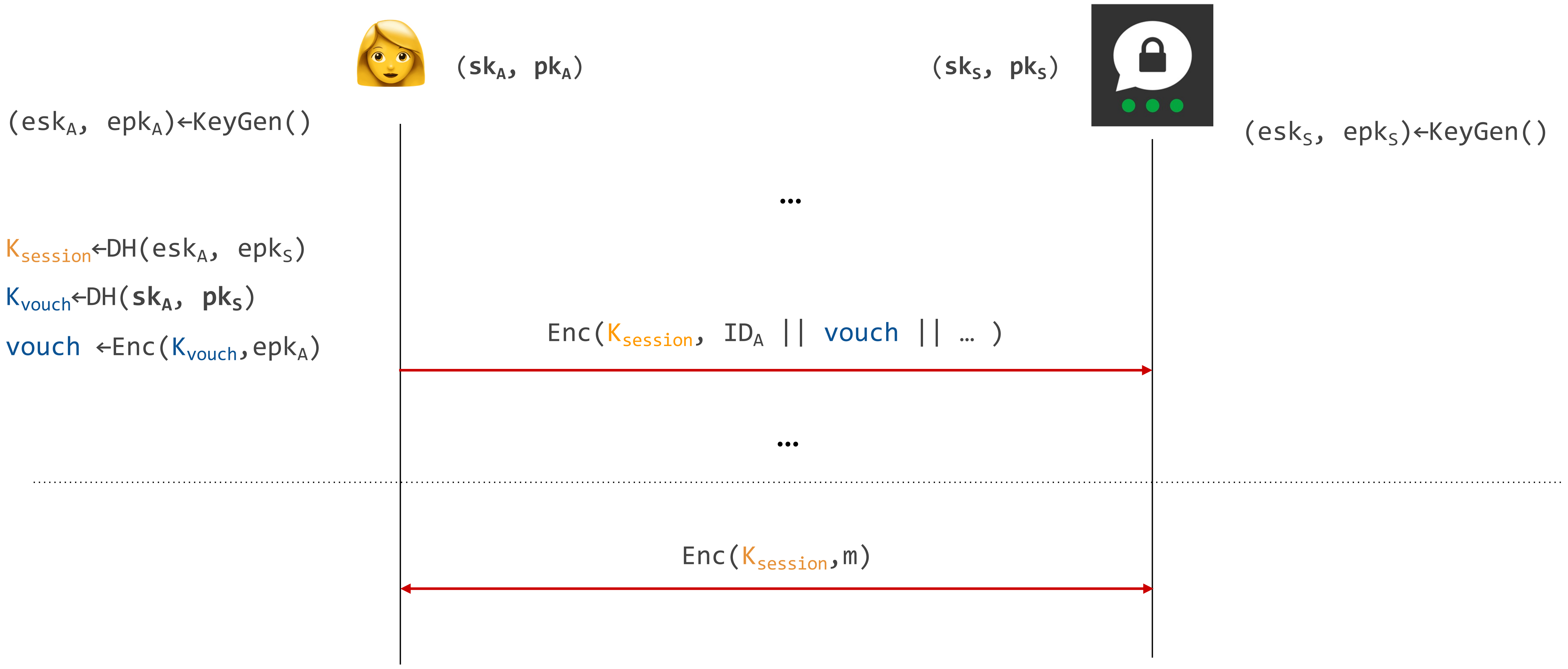
C2S Protocol



Establishes a client-server session key through an authenticated key exchange

C2S Protocol (simplified)

(Long-term keys **in bold font**)



$(\mathbf{sk_A}, \mathbf{pk_A})$

$(\mathbf{sk_S}, \mathbf{pk_S})$

$(esk_A, epk_A) \leftarrow KeyGen()$

$(esk_S, epk_S) \leftarrow KeyGen()$

…

$K_{session} \leftarrow DH(esk_A, epk_S)$

$K_{vouch} \leftarrow DH(\mathbf{sk_A}, \mathbf{pk_S})$

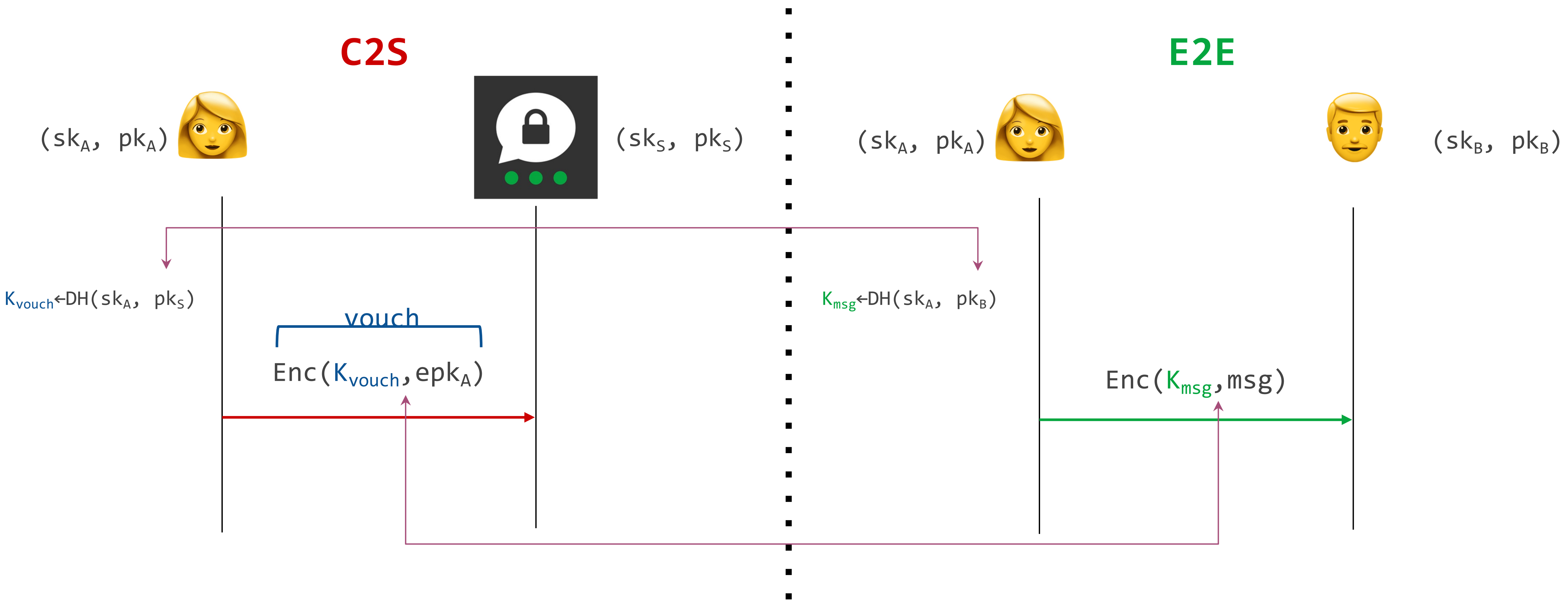
$vouch \leftarrow Enc(K_{vouch}, epk_A)$

$Enc(K_{session}, ID_A \,||\, vouch \,||\, \ldots)$

…

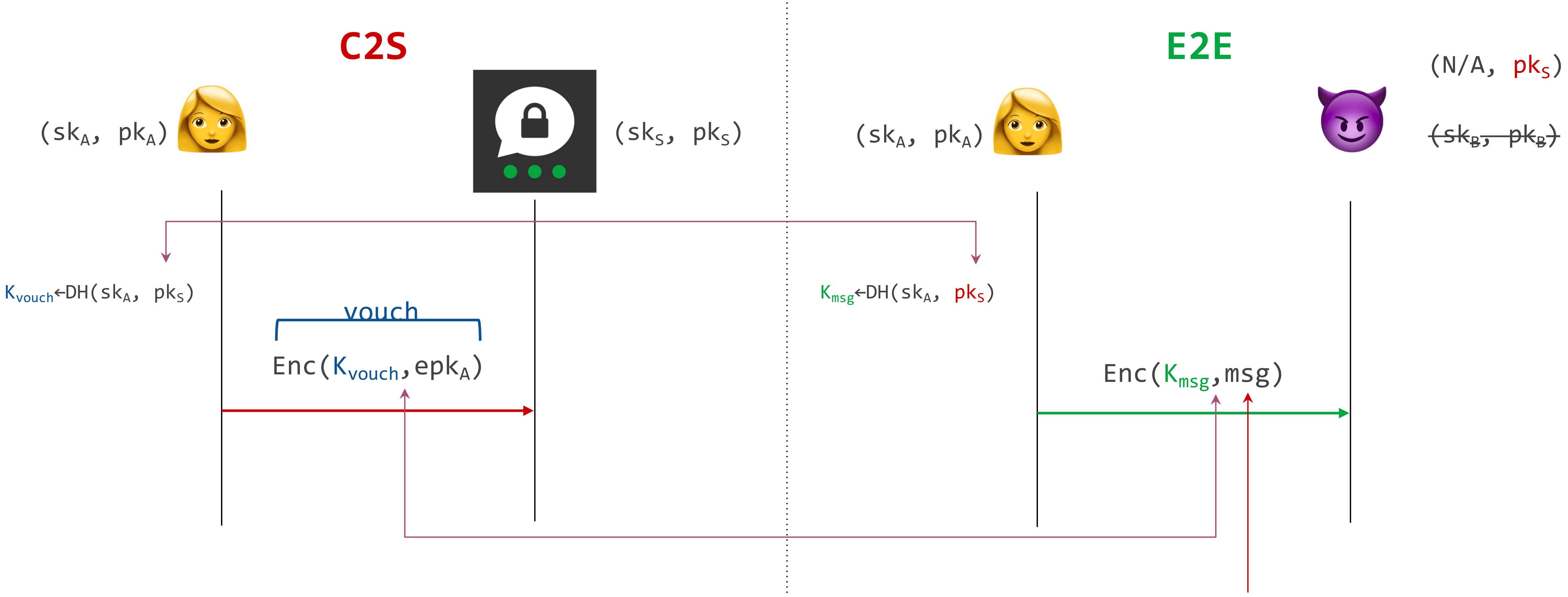
$Enc(K_{session}, m)$

Deja-vu?



C2S
E2E
$(sk_A,\ pk_A)$
$(sk_S,\ pk_S)$
$(sk_A,\ pk_A)$
$(sk_B,\ pk_B)$
$K_{vouch} \leftarrow DH(sk_A,\ pk_S)$
$\overbrace{Enc(K_{vouch}, epk_A)}^{vouch}$
$K_{msg} \leftarrow DH(sk_A,\ pk_B)$
$Enc(K_{msg}, msg)$

Deja-vu?

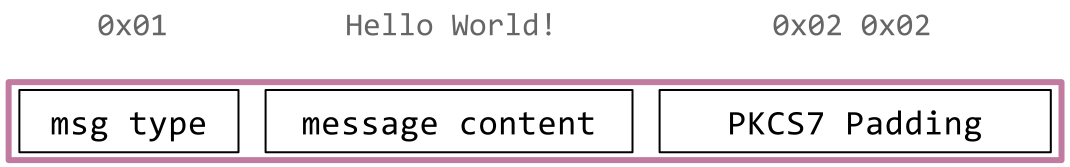


C2S
E2E
$(sk_A,\ pk_A)$
$(sk_S,\ pk_S)$
$(sk_A,\ pk_A)$
$(N/A,\ pk_S)$
~~$(sk_B,\ pk_B)$~~
$K_{vouch} \leftarrow DH(sk_A,\ pk_S)$
vouch
$\underbrace{Enc(K_{vouch}, epk_A)}$
$K_{msg} \leftarrow DH(sk_A,\ pk_S)$
$Enc(K_{msg}, msg)$

Attacker would like `msg` to encode a public key (for which it knows the corresponding private key)

Forging a Vouchbox

Recall: The structure of a message in the E2E protocol is

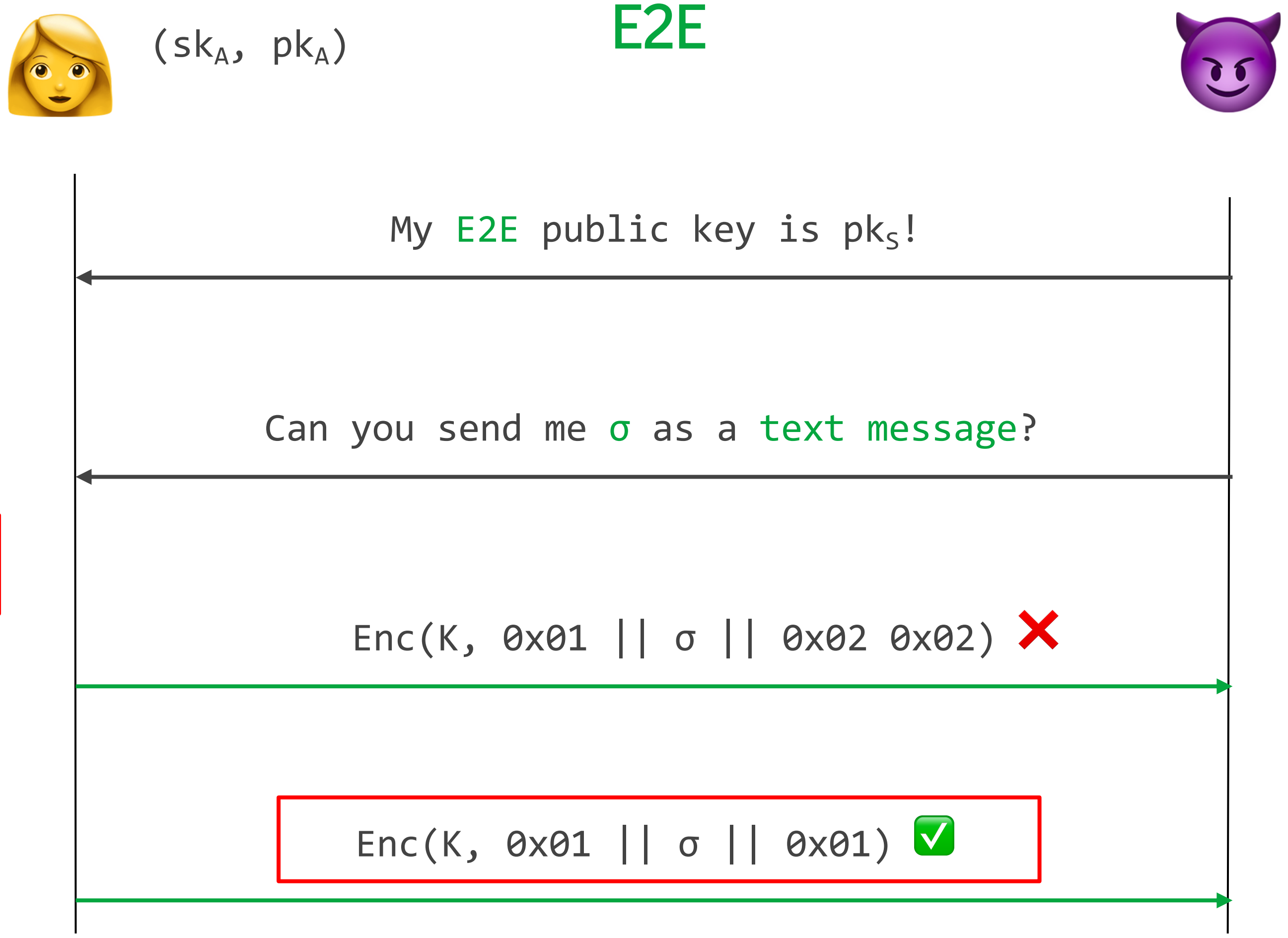


Idea: find a keypair `(esk, epk)` such that:

$$\texttt{epk} = \texttt{0x01} \;\|\; \boxed{\sigma} \;\|\; \texttt{0x01}$$

UTF-8 valid string of length 30B

Forging a Vouchbox



My E2E public key is pk_S!

Can you send me σ as a text message?

$\boxed{K \leftarrow DH(sk_A, pk_S)}$

$= K_{vouch}$ in C2S

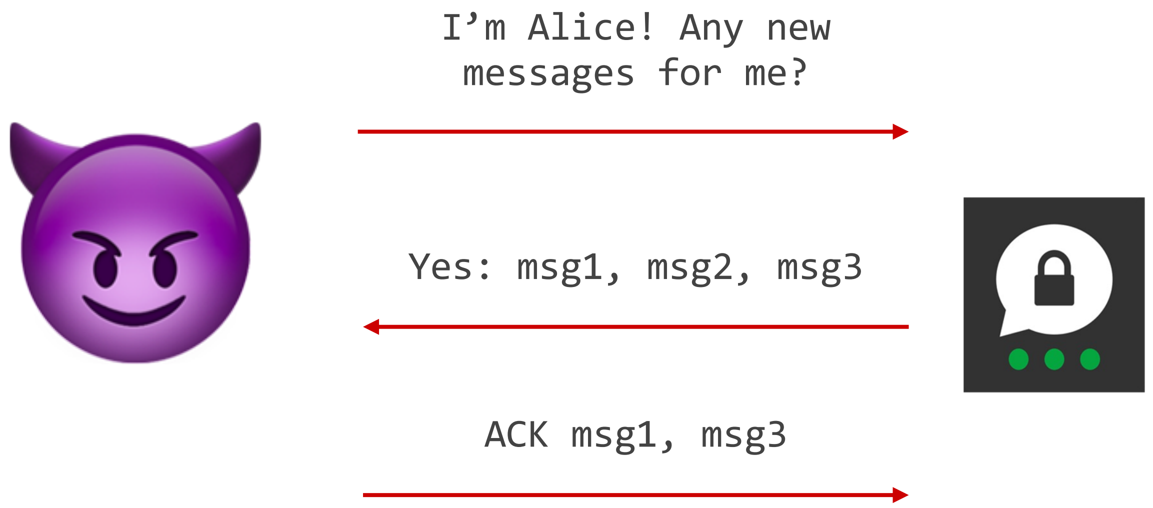
Enc(K, 0x01 || σ || 0x02 0x02) ✗

$\boxed{\text{Enc(K, 0x01 || σ || 0x01)}}$ ✅

$= \text{Enc}(K_{vouch}, \text{epk})$

Impact of Vouchbox Forgery

- Now the attacker can impersonate Alice to the server
- C2S x E2E cross-protocol attack
- Sending a text message… compromises client authentication forever!



Constructing the Keypair

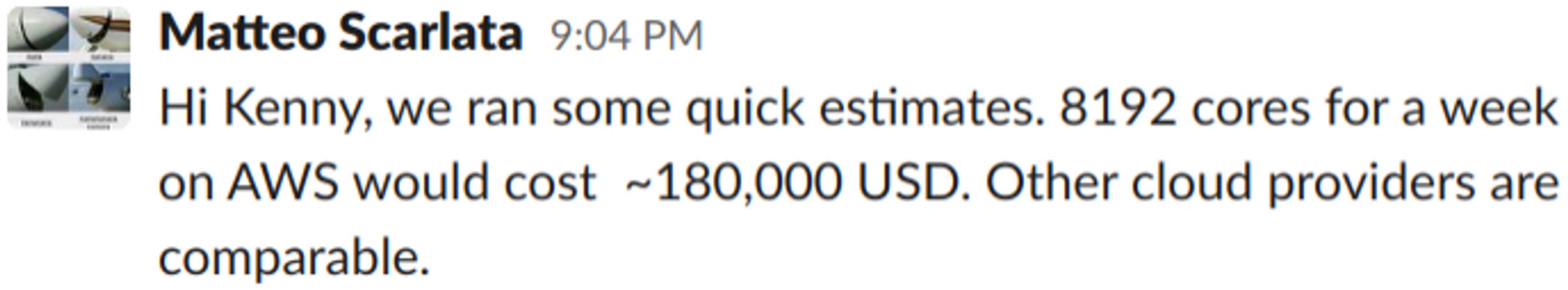
```
epk = 0x01 || σ || 0x01
```

(The σ is boxed.)

UTF-8 valid string of length 30B

Requires sampling 2^{51} keys!

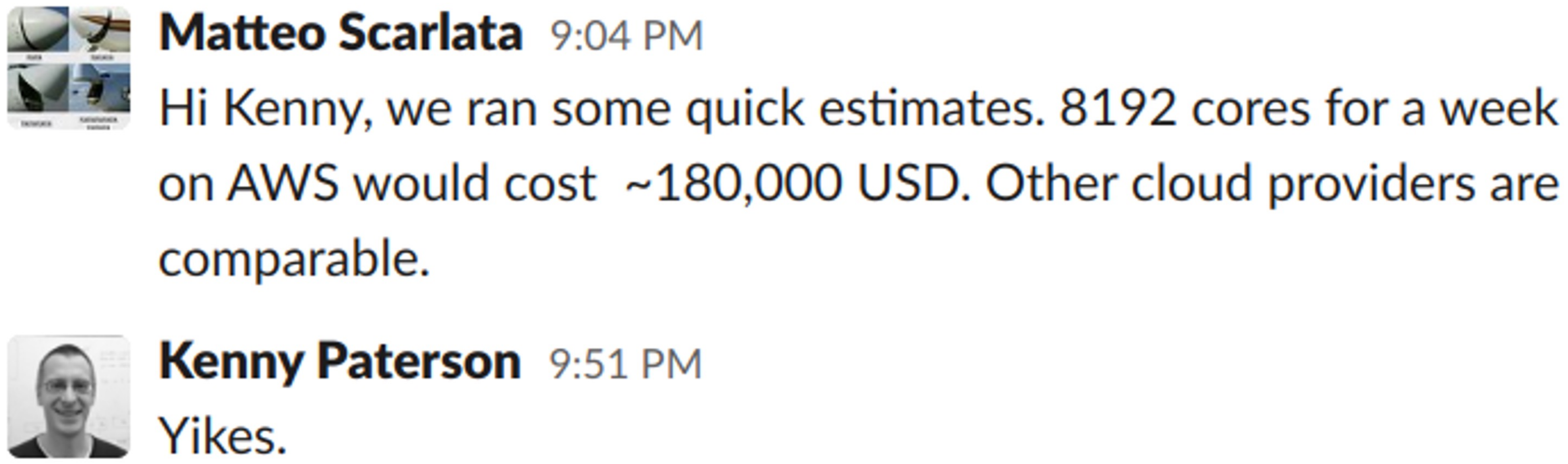
Constructing the Keypair



Matteo Scarlata 9:04 PM

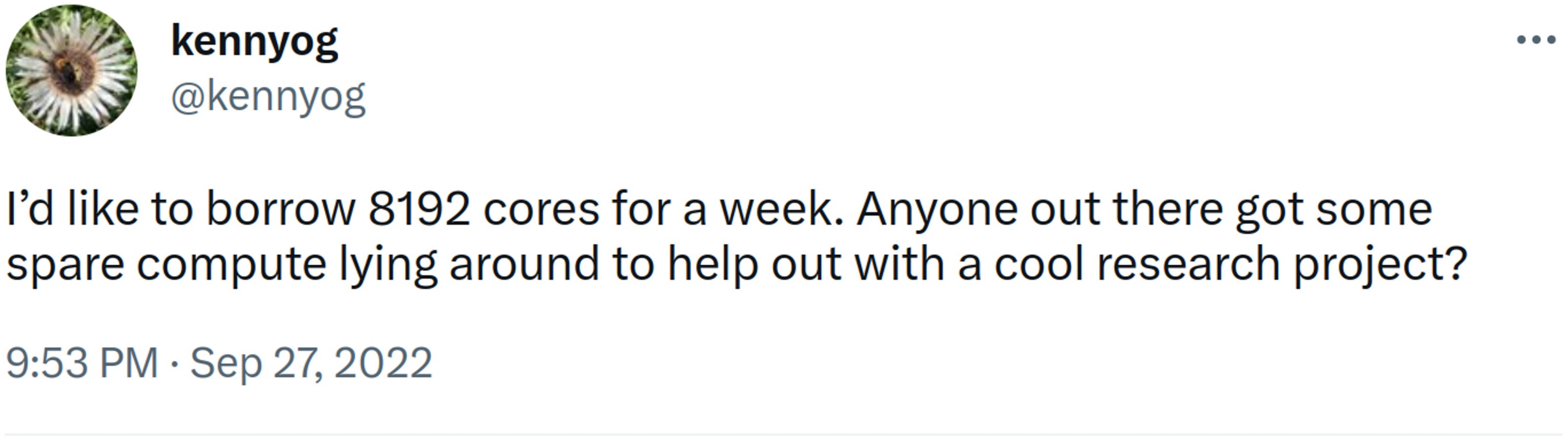
Hi Kenny, we ran some quick estimates. 8192 cores for a week on AWS would cost ~180,000 USD. Other cloud providers are comparable.

Constructing the Keypair


Matteo Scarlata 9:04 PM
Hi Kenny, we ran some quick estimates. 8192 cores for a week on AWS would cost ~180,000 USD. Other cloud providers are comparable.

Kenny Paterson 9:51 PM
Yikes.

Constructing the Keypair


kennyog
@kennyog

I'd like to borrow 8192 cores for a week. Anyone out there got some spare compute lying around to help out with a cool research project?

9:53 PM · Sep 27, 2022

Constructing the Keypair

Some optimizations and 8100 core-days later…

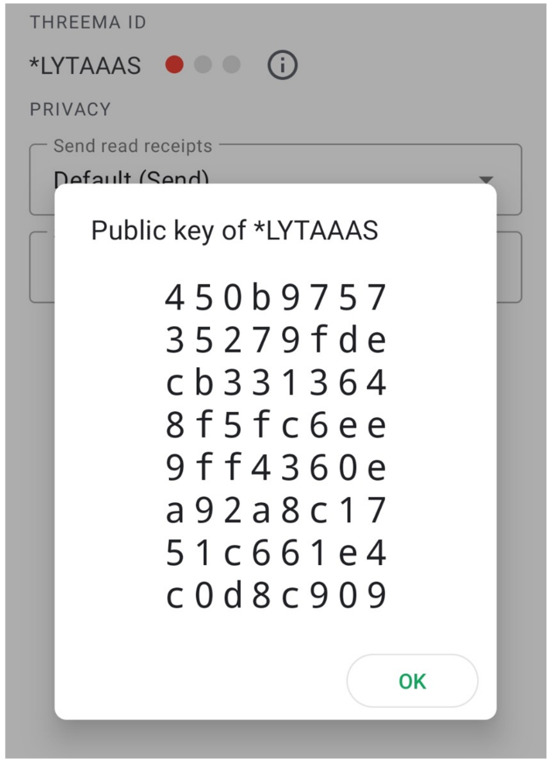
```
esk = 504ac13e00000000003000336d612d322d3232313231392d30332d303032300
0

epk  = 0175396a36df93276a6ae0a496d4bb5edf8331d79b573a2dcc813bdca1524101
```

u9j6ߓ'jjखԻ^߃1כW:-́;ܡRA

Registering the Server's Public Key

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

=> *LYTAAAS


THREEMA ID

*LYTAAAS

PRIVACY

Send read receipts

Default (Send)

Public key of *LYTAAAS

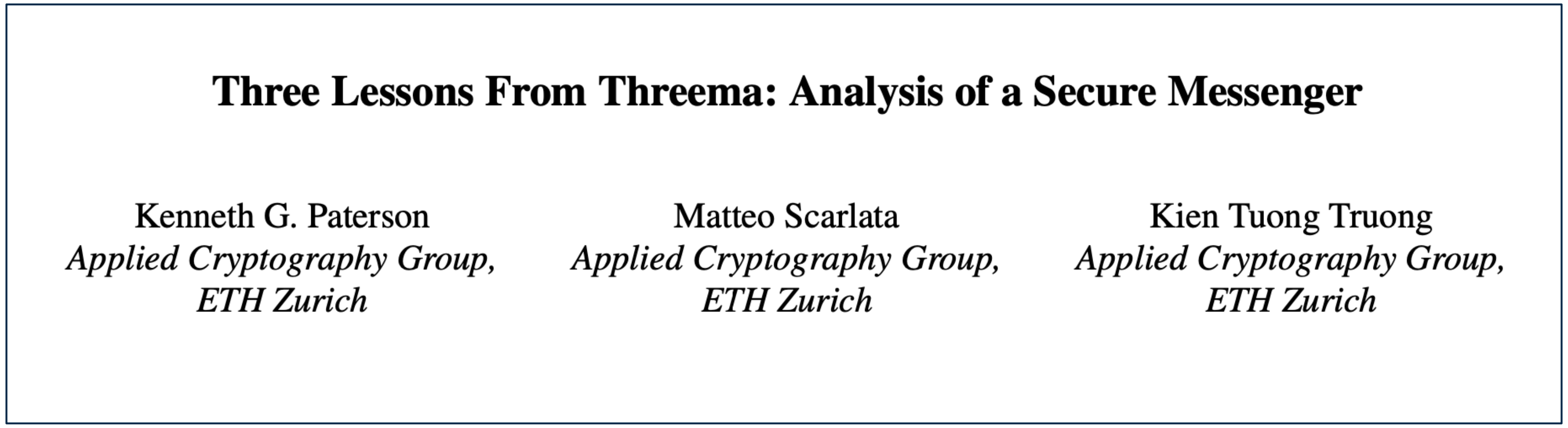
```
450b9757
35279fde
cb331364
8f5fc6ee
9ff4360e
a92a8c17
51c661e4
c0d8c909
```

OK

```
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
};
```

Threema – Wrap-up

- We've focussed on describing just one attack here.
- In total, we found 7 attacks (one of which was a rediscovery).
- Varying degrees of practicality and impact.
- Tricky disclosure and review process.



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

USENIX Security 2023, to appear
https://breakingthe3ma.app/

MEGA

- MEGA – E2EE cloud storage and communication platform with 280M registered users, 1000 Petabytes+ of stored data.
- Very strong security claims, promising users that MEGA cannot access user data.
- Backendal, Haller and Paterson [BHP23]: five attacks in malicious service provider setting.
- MEGA did not implement countermeasures proposed in [BHP23], instead relying on validation of plaintext payloads.
- These checks were sufficient to prevent the specific attacks of [BHP23], but not sufficient in general…

- **Albrecht, Haller, Marekova, Paterson [AHMP23]: two new attacks, also breaking confidentiality of user files.**

MEGA

The two new attacks of [AHMP23] are enabled by:

1. The continued lack of key separation and integrity protection in MEGA design.
2. An ECB encryption oracle present in MEGAdrop, a feature unrelated to the protocol under attack.
3. Detailed reporting of errors by the client to the server, added shortly after the patch for the attacks of [BHP23].

MEGA Key Hierarchy

Each user has:

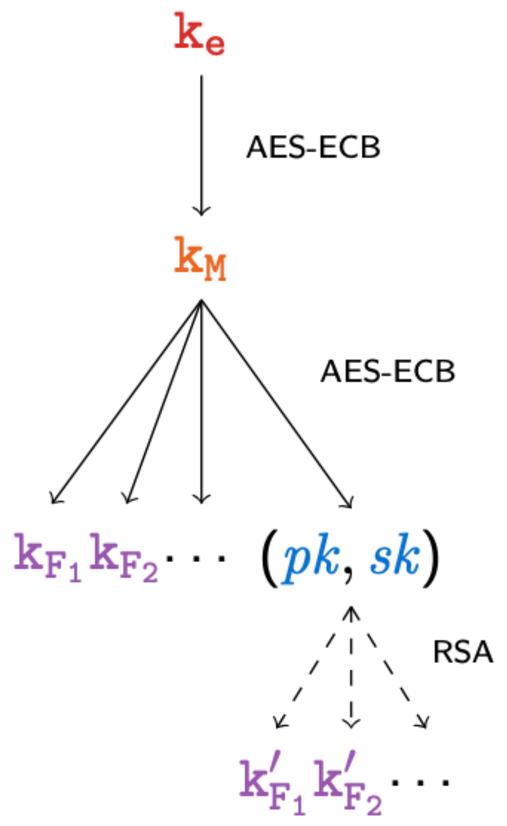
- a 128-bit *encryption key* k_e derived from their password
- a 128-bit *master key* k_M
- a 2048-bit RSA keypair (*pk*, *sk*) *file encryption keys* k_{F1} , k_{F2} , . . .

The keys are encrypted using AES-ECB via:

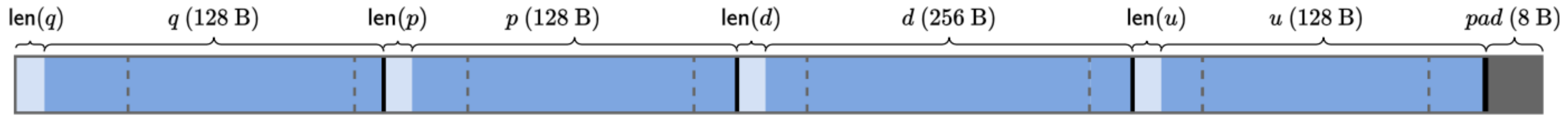
- $[k_M]_{ke}$
- $[k_F]_{kM}$,$[sk]_{kM}$

Shared-file encryption keys k'_F are encrypted under *pk* using RSA.

This key is also used in the user login/authentication protocol.



MEGA



- Custom encoding of *sk* for RSA-CRT decryption, referred to as privk:
  - the prime factors *p*, *q* of the RSA modulus.
  - the secret exponent *d* the value $u = q{-}1 \bmod p$.

- Each value is prefixed with a 2-byte length field
- Split into 16-byte blocks for AES-ECB

MEGA login procedure

User | *MEGA*

$$\xrightarrow{\text{login request}(\mathit{User})}$$

get $([\mathtt{k_M}]_{\mathtt{k_e}}, [\mathtt{privk}]_{\mathtt{k_M}}, \mathtt{uh})$ of *User*

pick 43-byte $\mathtt{sid}$

$[\mathtt{m}]_{pk} \leftarrow \text{RSA.Enc}(pk, \mathtt{sid} \parallel \mathtt{uh})$

$$\xleftarrow{([\mathtt{k_M}]_{\mathtt{k_e}}, [\mathtt{privk}]_{\mathtt{k_M}}, [\mathtt{m}]_{pk}, \mathtt{uh})}$$

$\mathtt{k_M} \leftarrow \text{AES-ECB.Dec}(\mathtt{k_e}, [\mathtt{k_M}]_{\mathtt{k_e}})$

$\mathtt{sid}' \leftarrow \textbf{MegaDec}(\mathtt{k_M}, [\mathtt{privk}]_{\mathtt{k_M}}, [\mathtt{m}]_{pk}, \mathtt{uh})$

$$\xrightarrow{\mathtt{sid}' \text{ or } \perp}$$

$\mathtt{sid}' \overset{?}{=} \mathtt{sid}$

MEGA client-side parsing and decryption

```
**MegaDec**($k_M$, $[privk]_{k_M}$, $[m]_{pk}$, uh):
    $sk$ ← DecryptPrivk($k_M$, $[privk]_{k_M}$) / AES-ECB
    $sid'$ ← DecryptSid($sk$, $[m]_{pk}$) / RSA-CRT
    Return $sid'$
```

Both steps rely on validity checking of the decrypted values and return distinguishable errors to the server!

MEGA client-side parsing and decryption

Explicit errors due to validity checking:

- In DecryptSid(sk,·), a length check on the plaintext together with a legacy padding check reveal if the second byte of *m* is 0.
- Yields an attack based on small subgroups.

Implicit errors due to bugs in the low-level library:

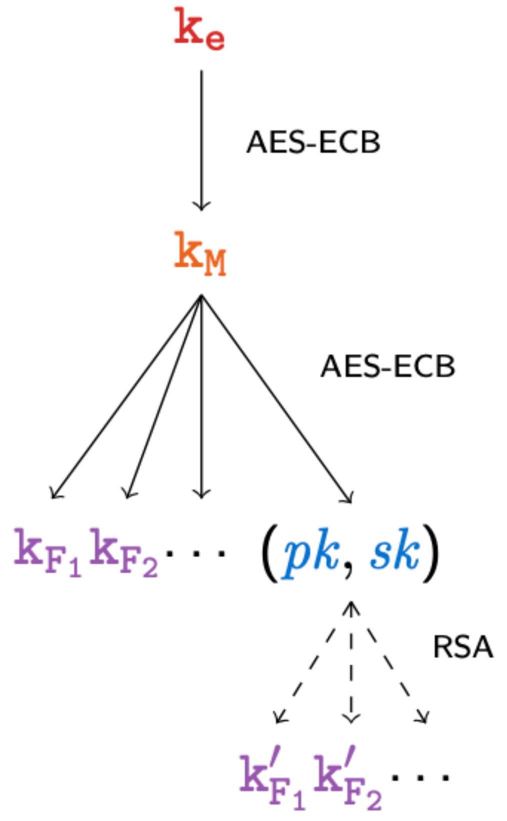
- In DecryptSid(k_M, ·), failure in recomputing $u' \leftarrow q^{-1} \bmod p$ reveals if $\gcd(p, q) \neq 1$.
- Yields an attack based on modular inverses.

Attacking MEGA

Threat model: malicious service provider trying to access customer data.

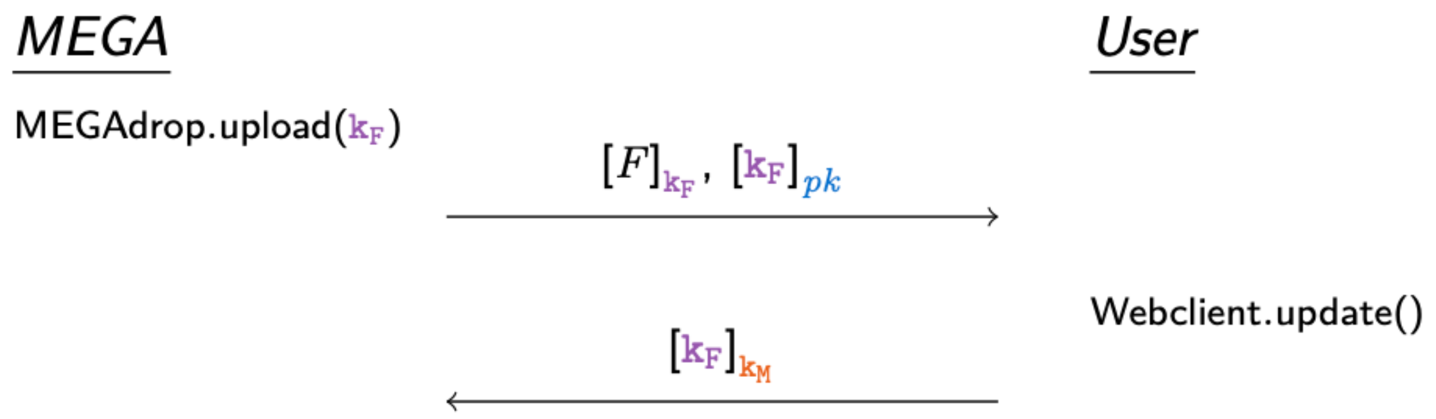
Goal: obtain ECB decryption capability under k_M in order to recover *sk* (or any k_F).

Cost measured mainly in the number of login attempts.



Warm-up: An ECB encryption oracle

- **MEGAdrop** lets anyone upload files to a folder in the cloud storage of the recipient.
- Clients automatically re-encrypt the received shared-file keys.


MEGA
User
MEGAdrop.upload($\mathtt{k_F}$)
$[F]_{\mathtt{k_F}}, [\mathtt{k_F}]_{pk}$
Webclient.update()
$[\mathtt{k_F}]_{\mathtt{k_M}}$

A malicious provider can construct an ECB encryption oracle *without user interaction* and *without leaving traces*!

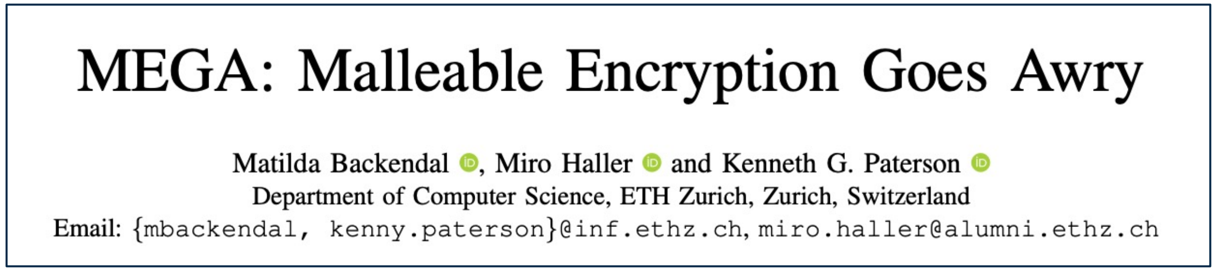
Attack Based on Modular Inverses

- Let $[B]_{kM}$ be the target ciphertext block, OECB_{kM} be the ECB encryption oracle, and $\perp_{inv}$ be the error output by **MegaDec** if $\gcd(p, q) \neq 1$.
- Main idea is for malicious server to construct $[\text{privk}^*]_{kM}$ (using OECB_{kM} and $[B]_{kM}$) such that
  - $p \bmod r = 0$ for small prime r, and
  - q contains B in the least-significant position
- Now $\perp_{inv}$ is output by **MegaDec** at client if and only if $q \bmod r = 0$.
- By adjusting q we can learn $B \bmod r$.
- Repeat for a set of primes r_i such that their product R is 128 bits.
- We can then learn B ($=B \bmod R$) from the $B \bmod r_i$ using CRT.
- Average cost: 627 login attempts and 66-91 OECB_{kM} queries.

MEGA – Disclosure and Remediation

- We disclosed to MEGA in September 2022, suggesting mitigations.
- MEGA informed us that they would work on a larger redesign meant to:
  - Change how private keys are stored,
  - Remove the ECB encryption oracle,
  - Replace the low-level crypto/bigint library.
- Upgrade of clients in March 2023.
- MEGA awarded a bug bounty and updated their security blog.
- Smart positioning by MEGA, portraying new attacks as part of previous set of attacks.
- Smooth disclosure and scientific review process.

Attacks Only Get Better…

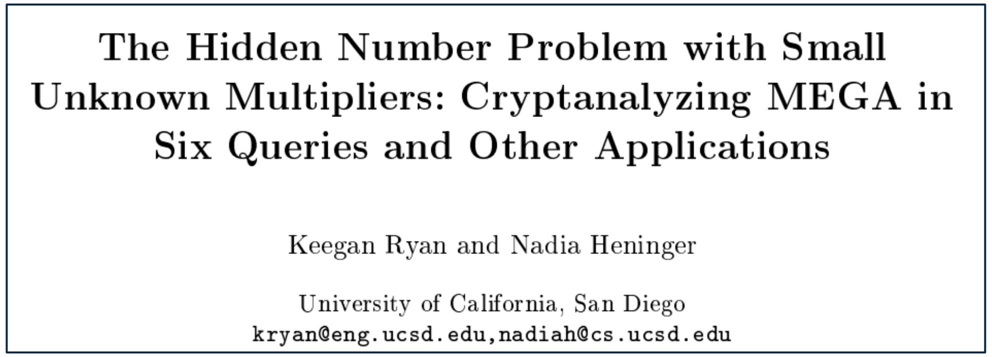


MEGA: Malleable Encryption Goes Awry

Matilda Backendal, Miro Haller and Kenneth G. Paterson
Department of Computer Science, ETH Zurich, Zurich, Switzerland
Email: {mbackendal, kenny.paterson}@inf.ethz.ch, miro.haller@alumni.ethz.ch

IEEE S&P 2023 🏆
https://mega-awry.io/

512 logins



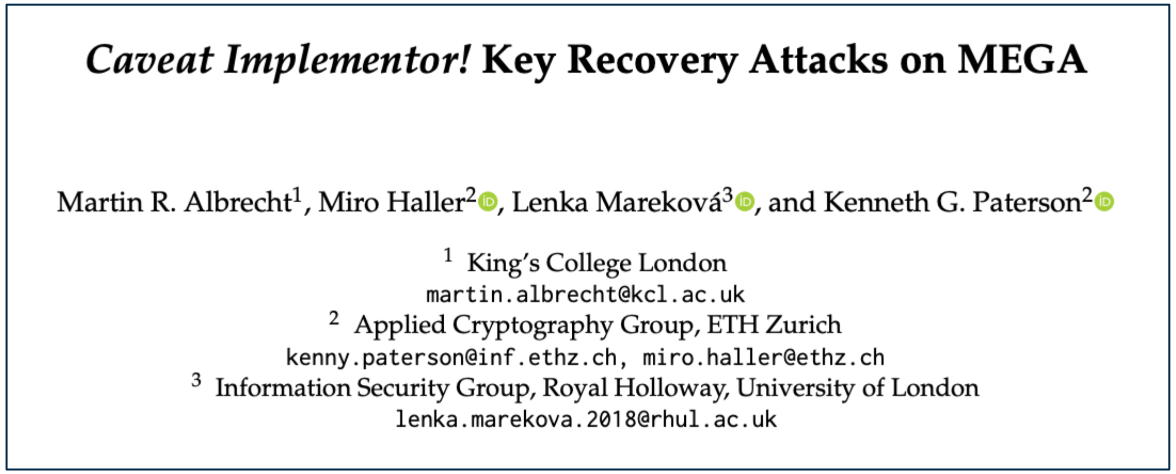
The Hidden Number Problem with Small Unknown Multipliers: Cryptanalyzing MEGA in Six Queries and Other Applications

Keegan Ryan and Nadia Heninger

University of California, San Diego
kryan@eng.ucsd.edu,nadiah@cs.ucsd.edu

PKC 2023 🏆
eprint.iacr.org/2022/914

6 logins
(on unpatched version)



Caveat Implementor! Key Recovery Attacks on MEGA

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

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

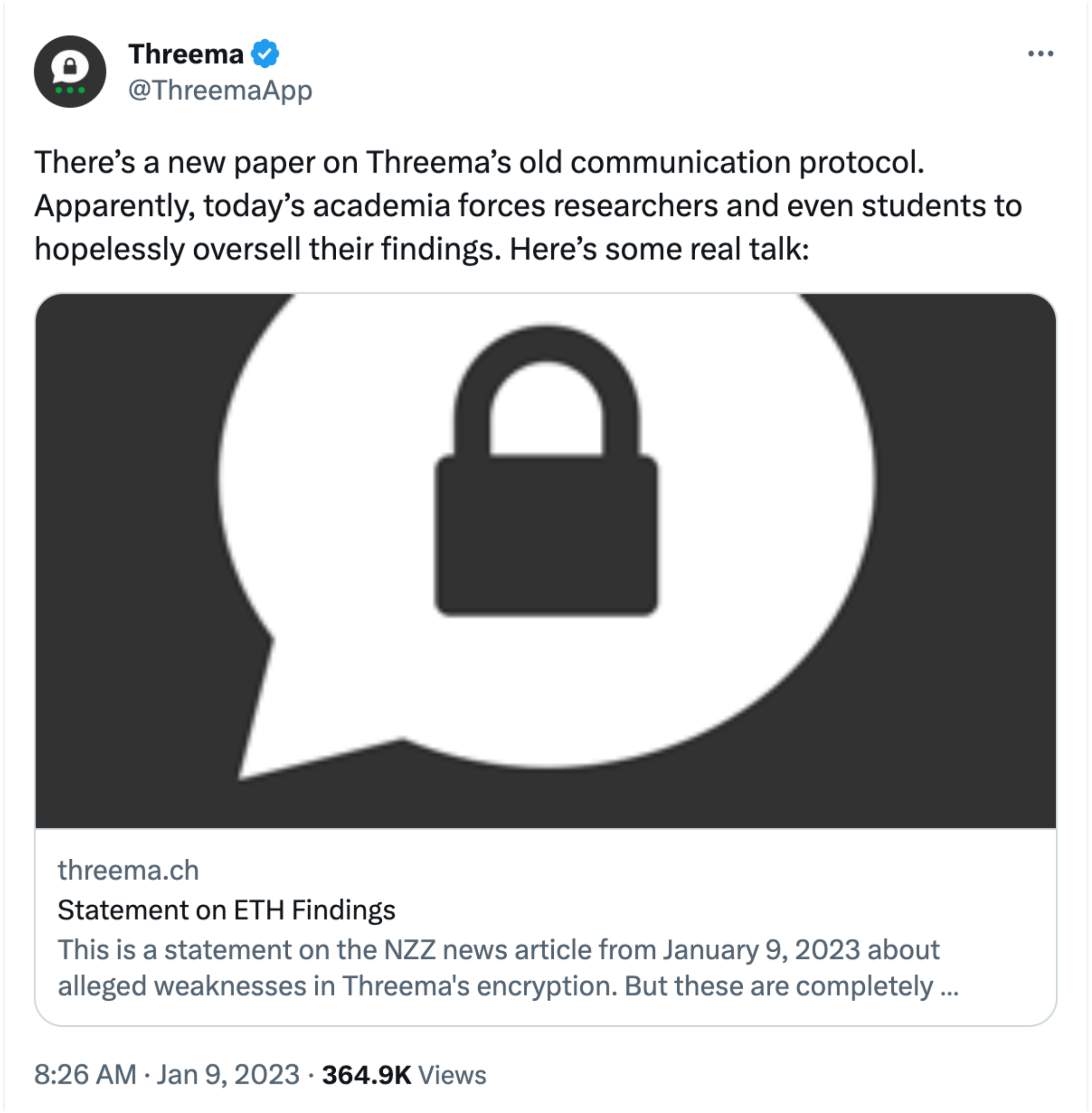
Eurocrypt 2023
mega-caveat.github.io/

2 logins
(on unpatched version)

Crypto in the Wild – Challenges

- Identifying all relevant cryptographic components and their interactions can be time-consuming.
- Attacks that work on paper may not work in practice due to unforeseen behaviours.
- Interacting with vendors can be complicated:
  - IND-CPA vs key recovery
  - Producing easy-to-use PoCs
  - CTO vs CEO
  - Lack of vendor experience in handling disclosure (e.g. uncoordinated patching)
  - Lack of willingness to engage in disclosure
- Interactions between disclosure and scientific process
- Extra work:
  - Disclosure
  - Website, logo, media,…

Crypto in the Wild – Challenges



Threema @ThreemaApp

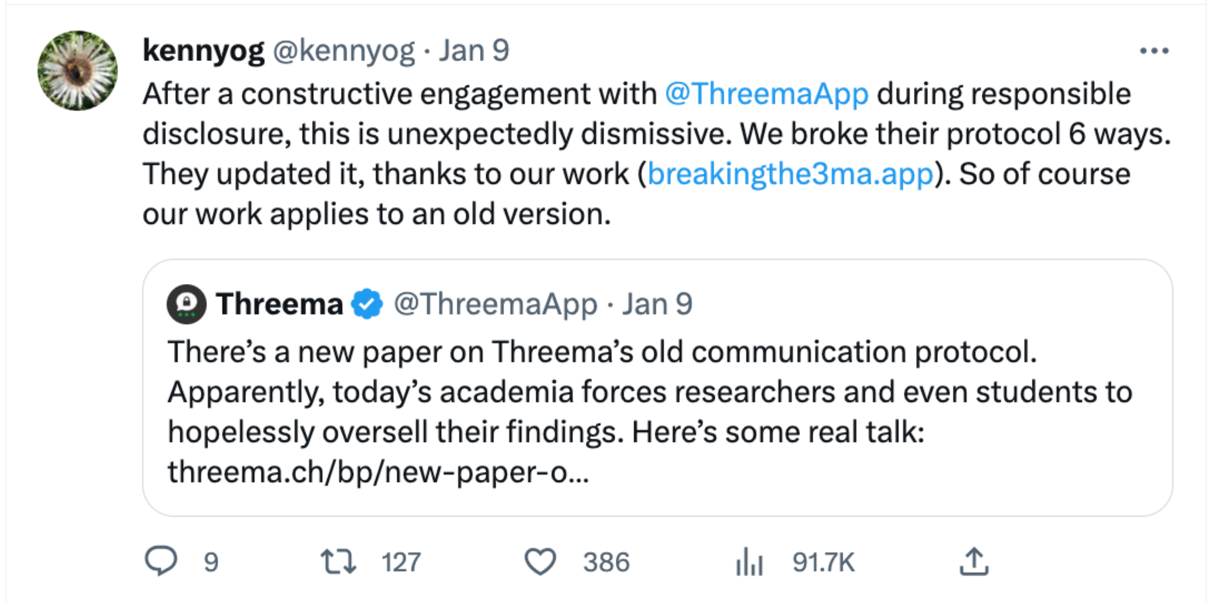
There's a new paper on Threema's old communication protocol. Apparently, today's academia forces researchers and even students to hopelessly oversell their findings. Here's some real talk:

threema.ch

Statement on ETH Findings

This is a statement on the NZZ news article from January 9, 2023 about alleged weaknesses in Threema's encryption. But these are completely ...

8:26 AM · Jan 9, 2023 · **364.9K** Views



kennyog @kennyog · Jan 9

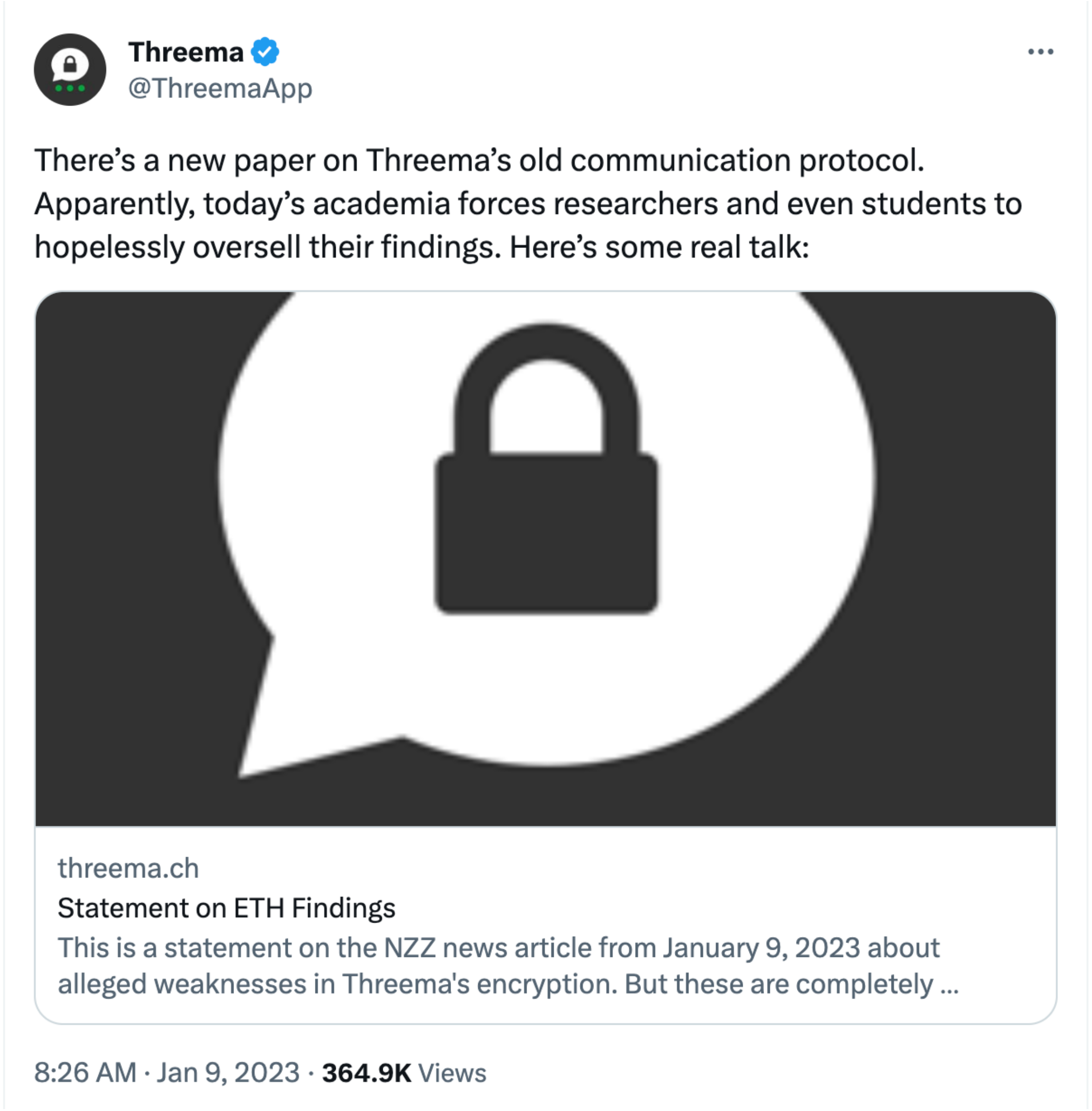
After a constructive engagement with @ThreemaApp during responsible disclosure, this is unexpectedly dismissive. We broke their protocol 6 ways. They updated it, thanks to our work (breakingthe3ma.app). So of course our work applies to an old version.

Threema @ThreemaApp · Jan 9

There's a new paper on Threema's old communication protocol. Apparently, today's academia forces researchers and even students to hopelessly oversell their findings. Here's some real talk: threema.ch/bp/new-paper-o...

9 127 386 91.7K

Crypto in the Wild – Challenges

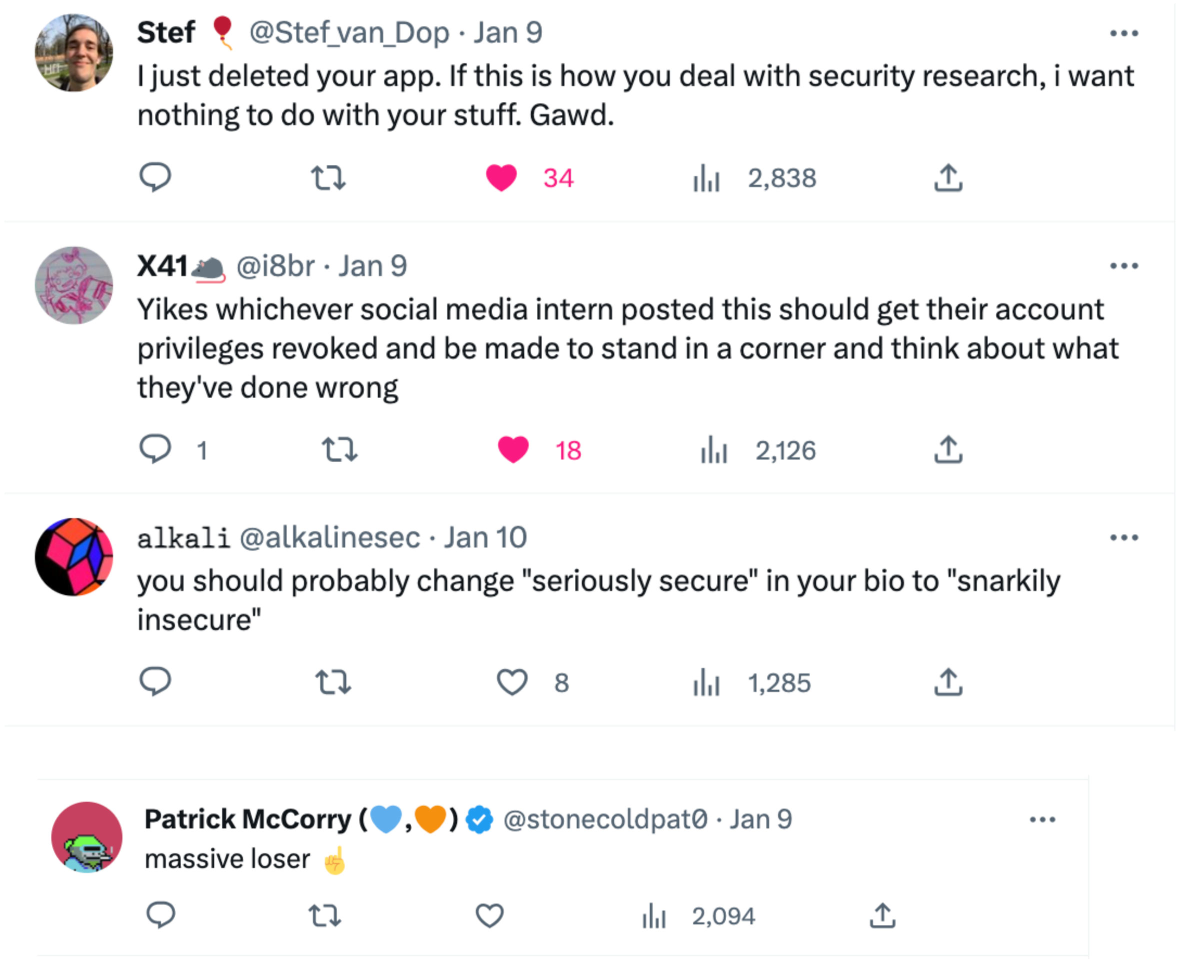


Threema @ThreemaApp

There's a new paper on Threema's old communication protocol. Apparently, today's academia forces researchers and even students to hopelessly oversell their findings. Here's some real talk:

threema.ch
Statement on ETH Findings
This is a statement on the NZZ news article from January 9, 2023 about alleged weaknesses in Threema's encryption. But these are completely ...

8:26 AM · Jan 9, 2023 · **364.9K** Views



Stef @Stef_van_Dop · Jan 9
I just deleted your app. If this is how you deal with security research, i want nothing to do with your stuff. Gawd.
34 2,838

X41 @i8br · Jan 9
Yikes whichever social media intern posted this should get their account privileges revoked and be made to stand in a corner and think about what they've done wrong
1 18 2,126

alkali @alkalinesec · Jan 10
you should probably change "seriously secure" in your bio to "snarkily insecure"
8 1,285

Patrick McCorry (💙,🧡) @stonecoldpat0 · Jan 9
massive loser ☝️
2,094

Crypto in the Wild – A Word on Audits

- Many of the systems we analysed had already been subjected to cryptographic audits.
- 6 months of a determined ETH Master’s student plus research team compared to a few days of work by a bored auditor.
- You (sometimes) get what you pay for.

Crypto in the Wild – The Why

- It’s fun!
- You may find novel attack vectors that are applicable elsewhere.
- It’s a great tool for engaging students in the field.
- It provides teachable moments to use in the classroom.
  - Key re-use really is bad.
  - Integrity protection really does matter.
  - etc.
  - (But this is only relevant if you teach cryptography in a particular way.)

Crypto in the Wild – The Why

APPLIED CRYPTO GROUP

- Cryptography has become ubiquitous and crucial to the security of the digital society.
- Theory is necessary, but not sufficient, for building secure systems.
- “Crypto in the Wild” research plays an important role in bridging the gap between theory and practice.
  - Identifying new use cases and cryptographic needs not currently met by what’s available in the literature.
  - Identifying how and why developers get cryptography wrong.

Crypto in the Wild – The Why Not

- Extra effort beyond the usual scientific process.
- Disclosure may go wrong and can generate adverse publicity.
- Much of the low-hanging fruit may already have been harvested. (??)
- Resulting research papers tend not to be highly cited. (??)
    - But do seem to win best paper awards. ☺

BuT iS It sCiENce?

- Sometimes observed in paper reviews…
- If science is about the study and understanding of natural phenomena, then, yes, it is science.
- How developers use and abuse cryptography may tell us something useful about:
  - Our education system (how we teach cryptography).
  - How we make cryptography available to developers (developer-proof libraries).
  - The theory/practice pipeline.

Future Perspective

- "Crypto in the Wild" has been successful in the traditional areas of cryptographic application: first secure communications, then data at rest.
- But cryptography is starting to be widely used for much more advanced applications, including "data under computation" (MPC, FHE, ZKP) as well as in cryptocurrencies.
- Based on our work so far, and work of others, we predict that the mistakes of the past will be repeated in these new settings:
  - Issues in the core algorithms and crypto-libraries.
  - Mis-application of the libraries in building secure protocols and systems.
  - Example: Ethereum PoW sync mechanisms.
- Let's collaborate!

ETH zürich

Contact:

Professor Kenny Paterson
Applied Cryptography Group
kenny.paterson@inf.ethz.ch

ETH Zurich
Applied Cryptography Group
Department of Computer Science
Universitätstrasse 6
8092 Zurich, Switzerland

https://appliedcrypto.ethz.ch/

APPLIED
CRYPTO
GROUP