Facilis Descensus:
Why Threema Failed in Practice

Kenny Paterson, **Matteo Scarlata**, Kien Tuong Truong

Based on "Analysis of the Threema Secure Messenger"
https://www.usenix.org/conference/usenixsecurity23/presentation/paterson

ETH zürich
APPLIED CRYPTO GROUP

What is Threema?

- An “end-to-end encrypted instant messaging application” for Android and iOS
- 11 million private users worldwide

Threema.

Schweizerische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confederaziun svizra

Mercedes-Benz

DEUTSCHE BUNDESBANK
EUROSYSTEM

Emirates

Lamest Vendor Response 2023 goes to...

Threema.

How did we get here?

Attack: C2S Ephemeral Key Compromise

Attack: Vouch Box Forgery

Attack: Message Reordering/Omission

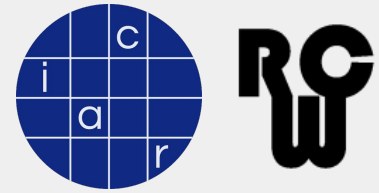
Attack: Message Replay/Reflection

Attack: Kompromat

Attack: Compression-Side Channel on Threema Safe

Attack: Threema ID Export

Technical Talks



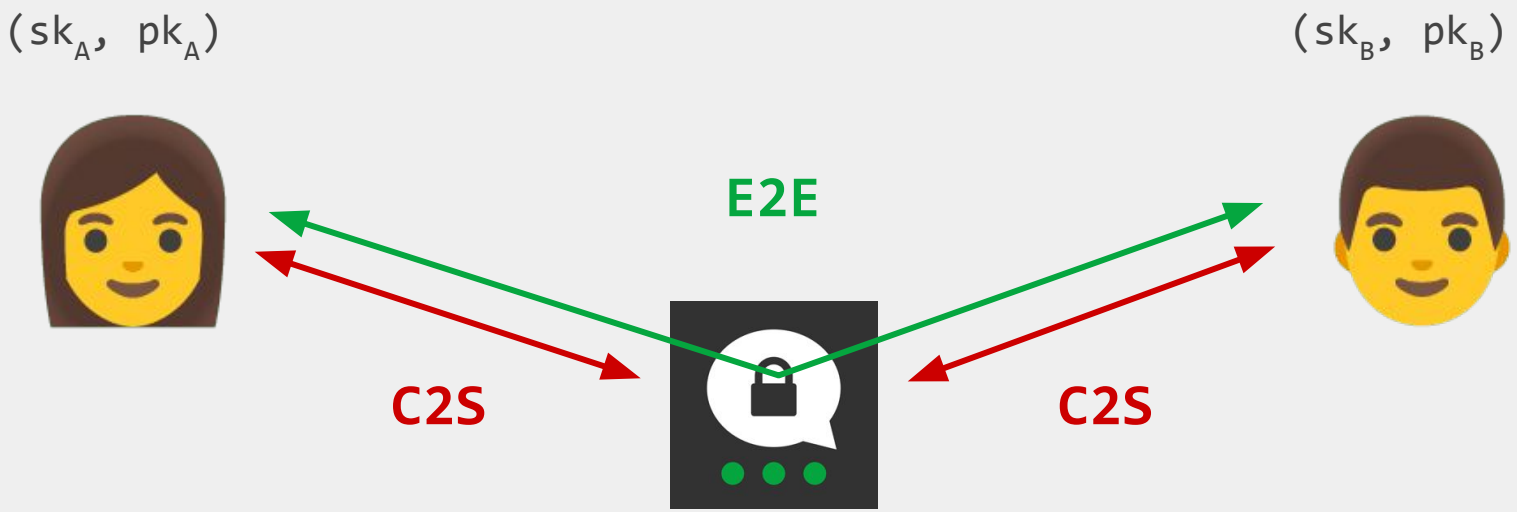
https://youtu.be/sthXs4zJ5XU?t=1892

usenix
THE ADVANCED
COMPUTING SYSTEMS
ASSOCIATION

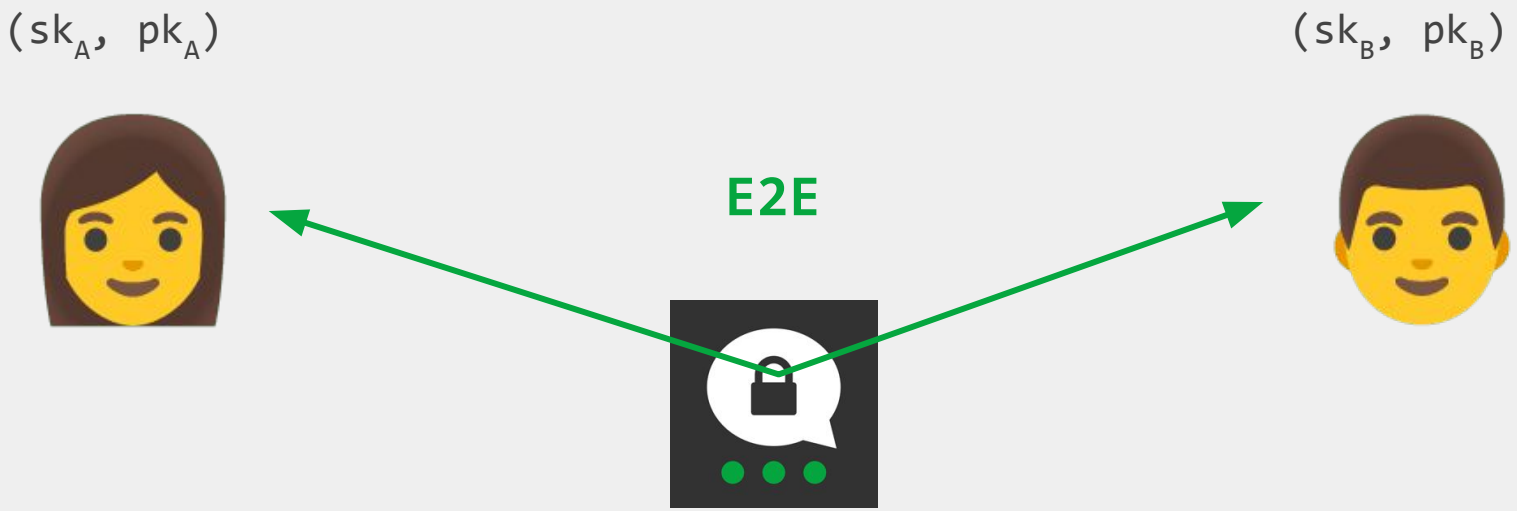
https://www.usenix.org/conference/
usenixsecurity23/presentation/paterson

2012

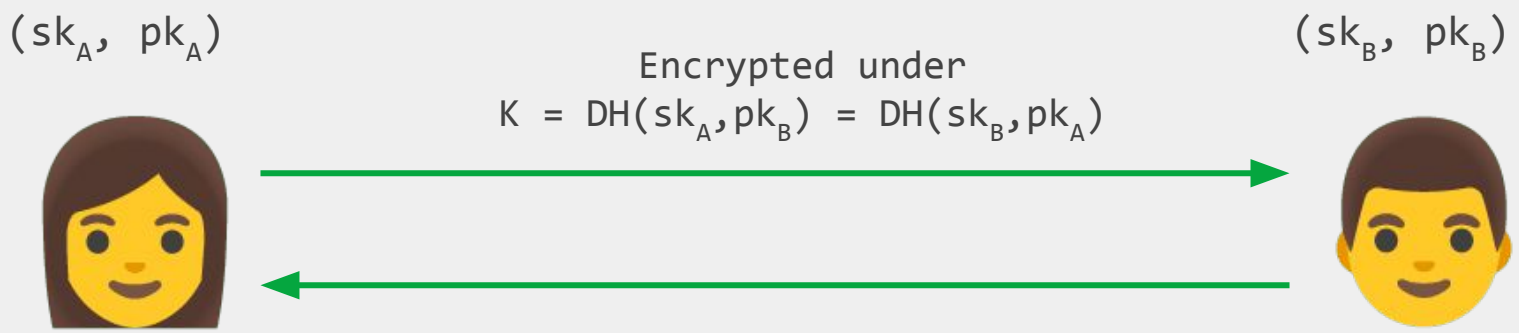
What do we want?



What we want? End-to-End Encryption!

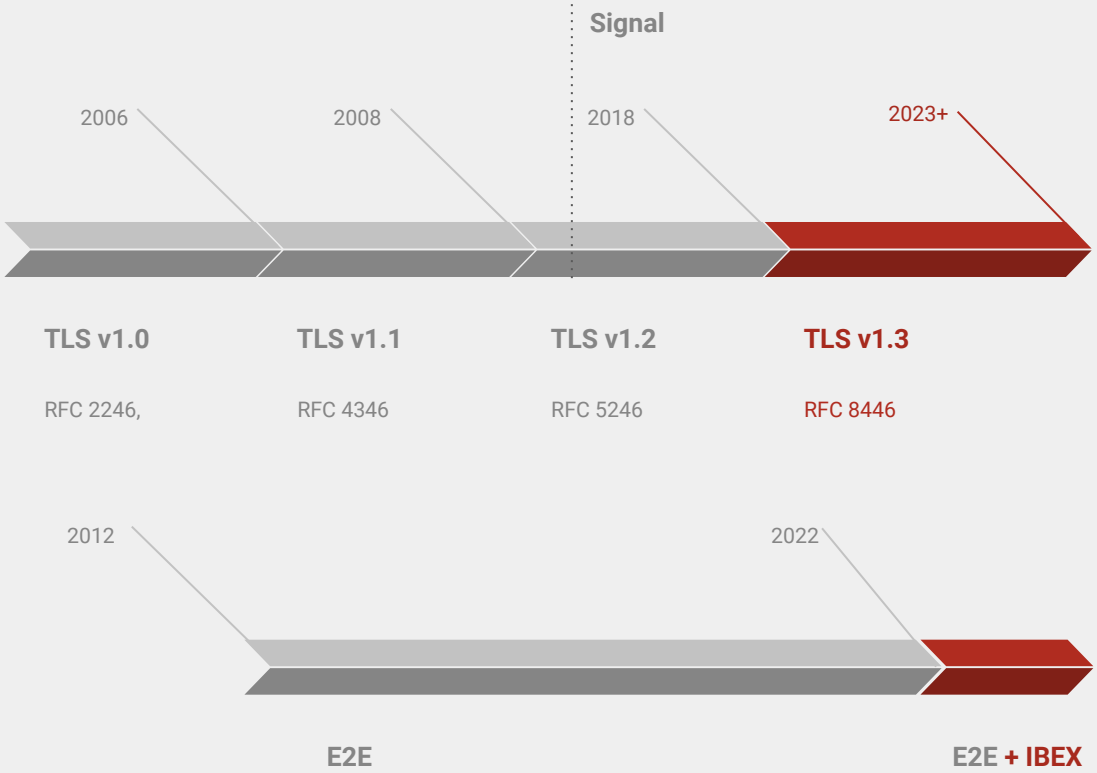


E2E Protocol



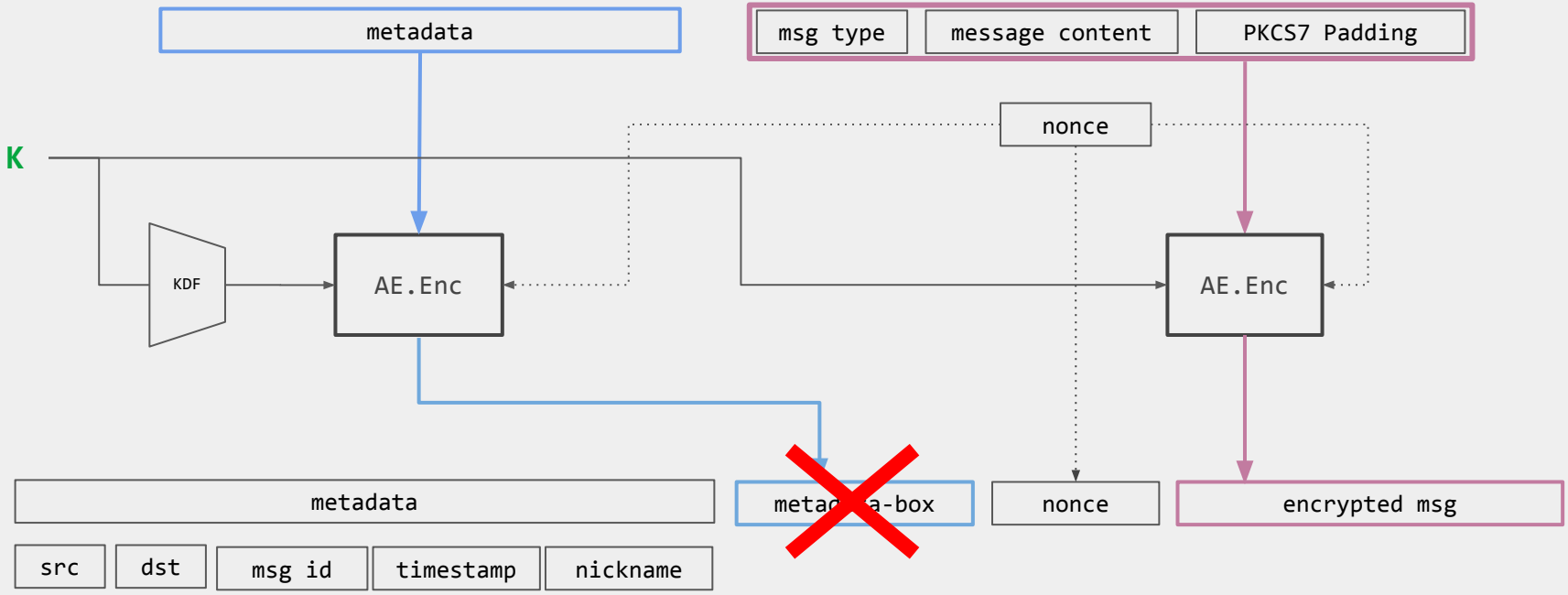
No Forward Secrecy!

What went wrong?



No incentives to update!

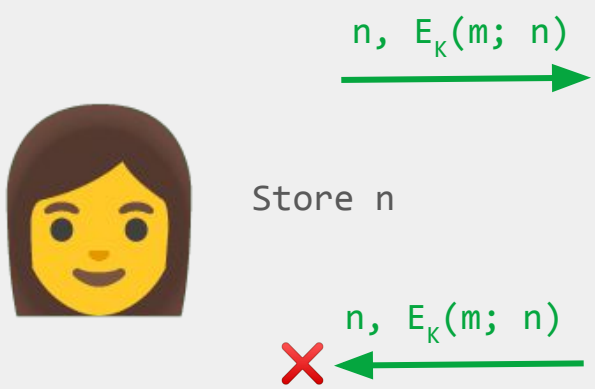
E2E Protocol: Message Structure



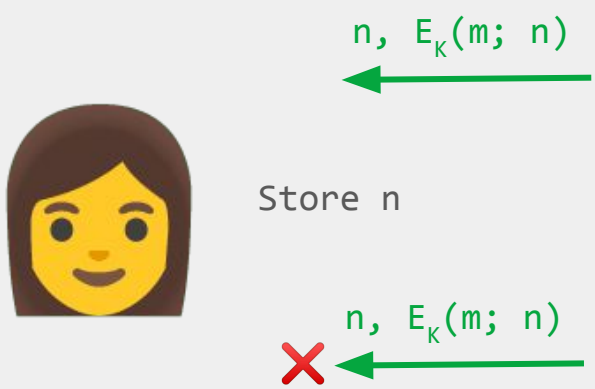
No metadata integrity!

E2E: Saving nonces

Reflection Attacks



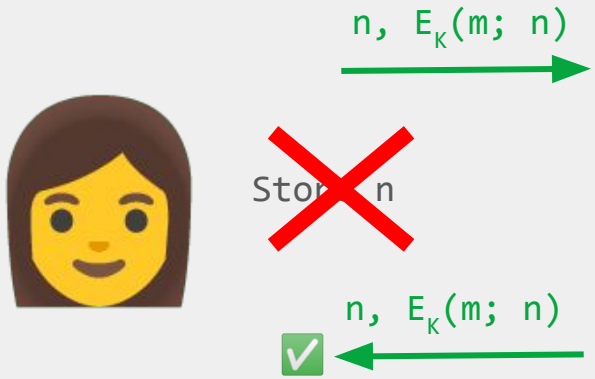
Replay Attacks



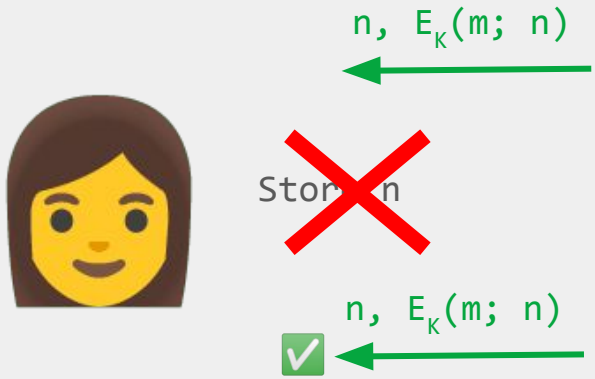
In both cases, reject, since n was already seen

E2E: Saving nonces

Reflection Attacks



Replay Attacks



Messages can be reflected/replayed!

What went wrong?

NaCl

08 Sep 2020

NACL IS NOT A HIGH-LEVEL API

When talking about high-level application cryptography APIs I usually hear mentioned libsodium, Tink, pyca/cryptography, and NaCl.

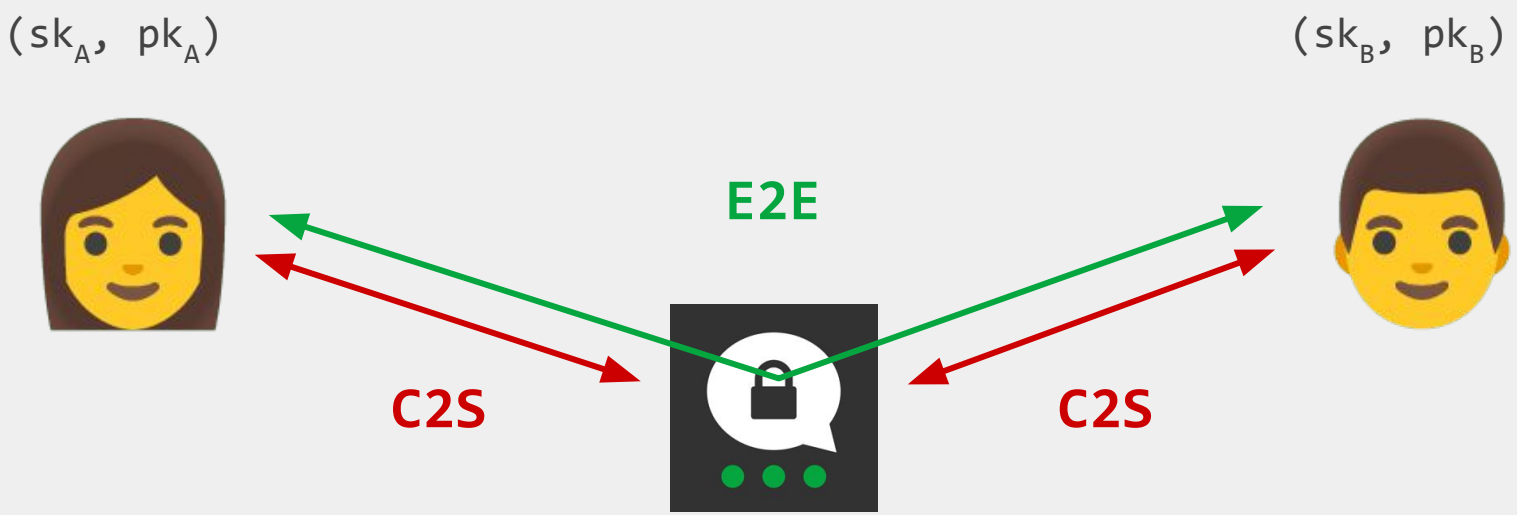
One of these things is not like the others! The value NaCl had 10 years ago was that it was an opinionated library at a time when all cryptography libraries were choose-your-own-adventure toolkits, but its APIs are not high-level, and even its constructions are unsafe by today's standards.

API does not support AD!

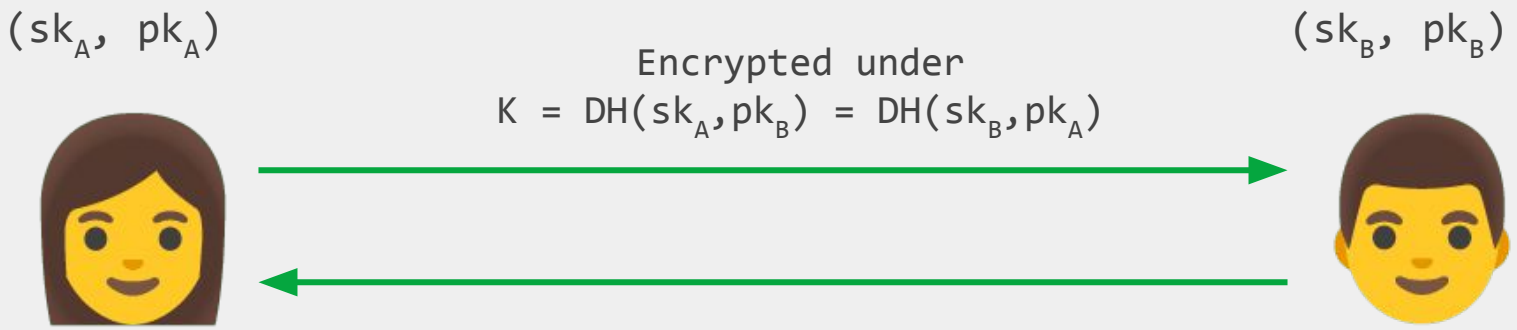
Backwards compatibility

Bad crypto design!

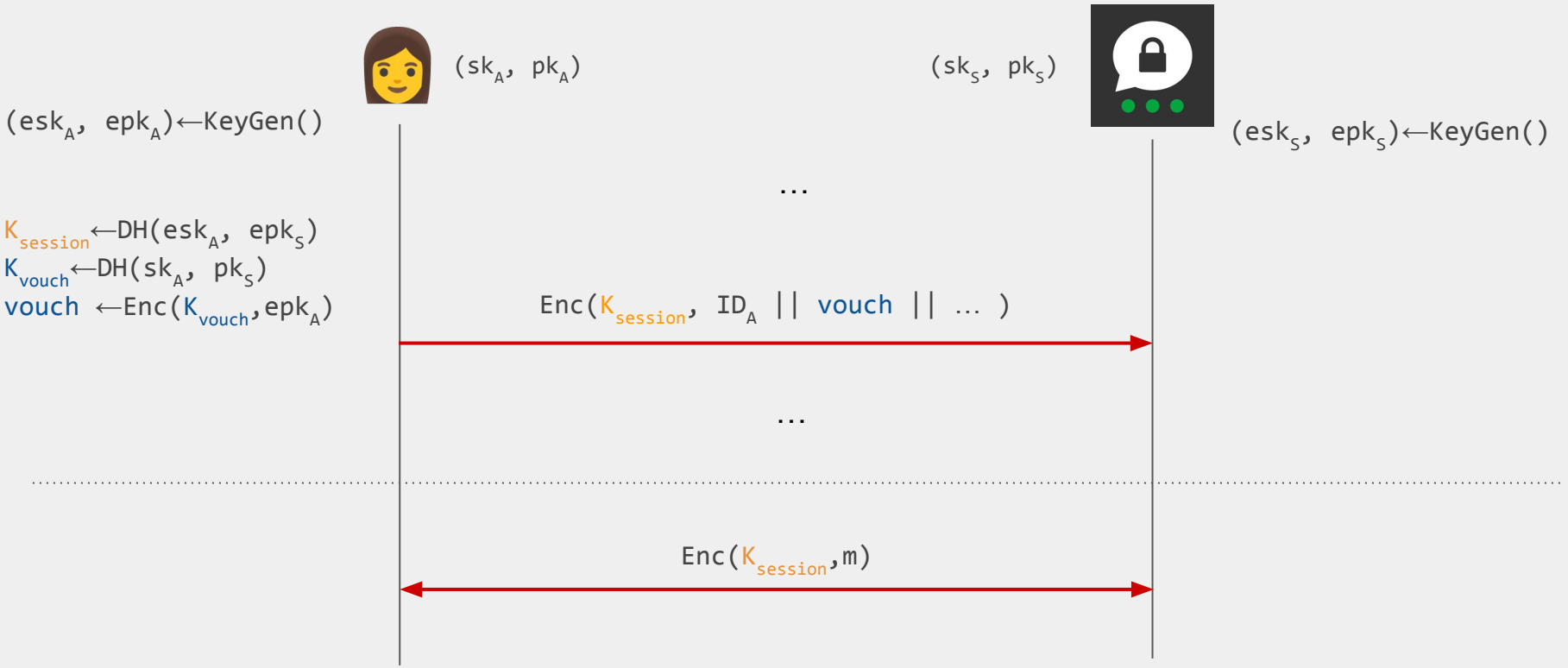
What do we want?



E2E Protocol

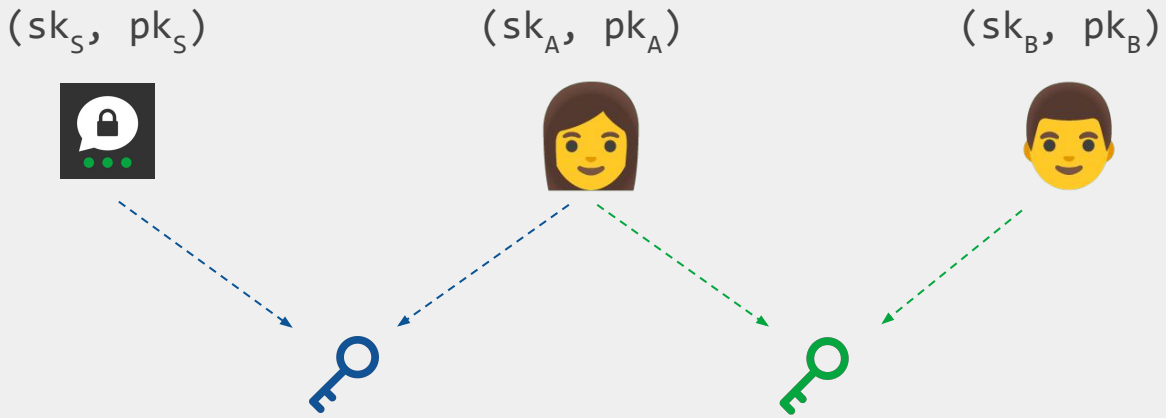


C2S Protocol*

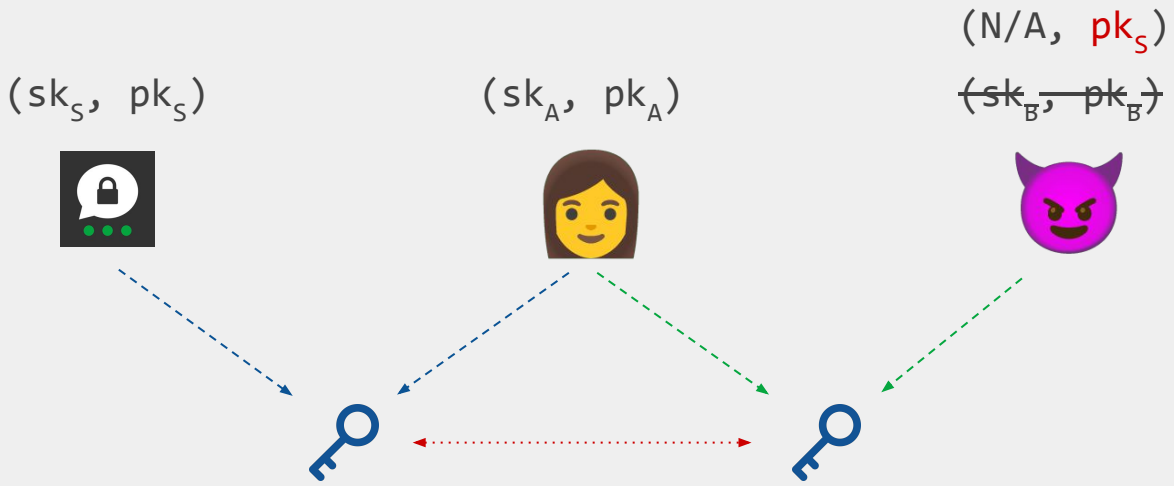


* Simplified, details omitted

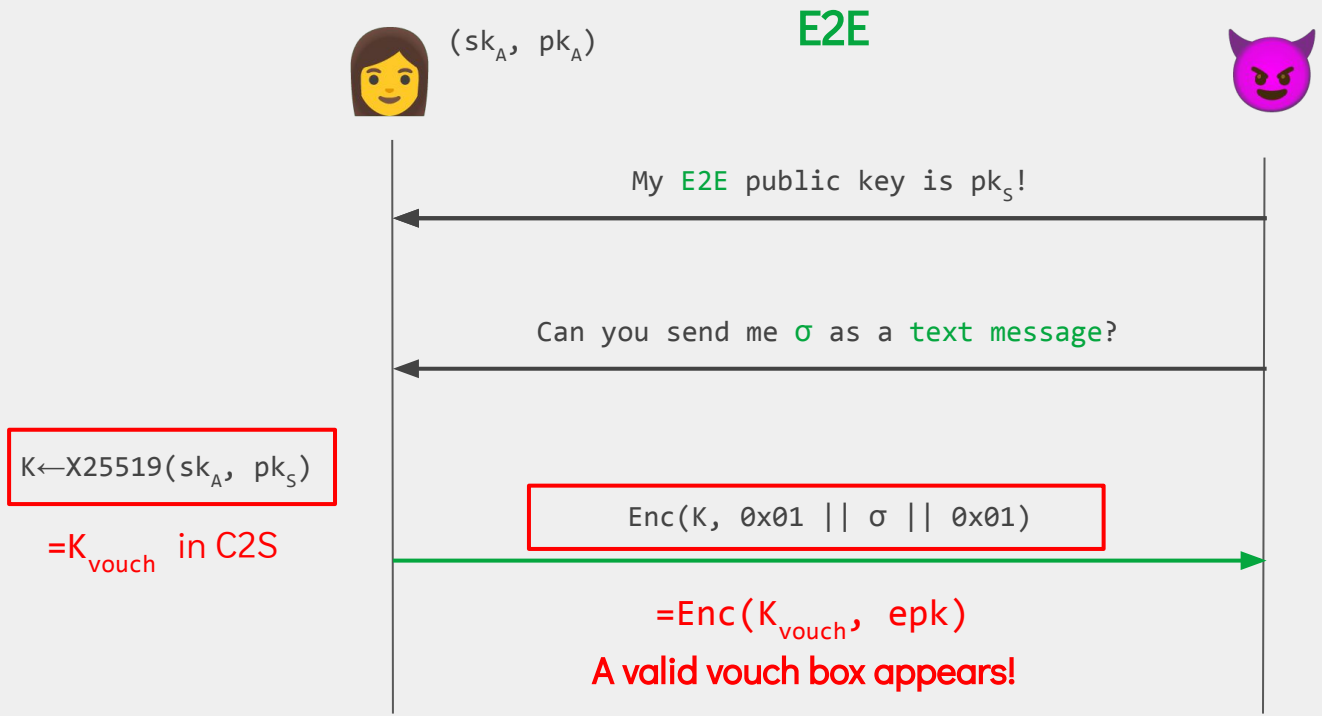
Deja-vu?



Deja-vu?



Cross-Protocol Attack



What went wrong?

NaCl

```
#include "crypto_box.h"

std::string pk;
std::string sk;
std::string n;
std::string m;
std::string c;

c = crypto_box(m,n,pk,sk);
```

pk_A, sk_A <- KeyGen()
pk_B, sk_B <- KeyGen()

n <-$- N

c = Enc(m, n, pk_A, sk_B)
  = SK.Enc(m, n, DH(pk_A, sk_B))

NaCl

```
#include "crypto_box.h"

std::string pk;
std::string sk;
std::string n;
std::string m;
std::string c;

m = crypto_box_open(c,n,pk,sk);
```

pk_A, sk_A <- KeyGen()
pk_B, sk_B <- KeyGen()

m = Dec(c, n, pk_B, sk_A)
  = SK.Dec(c, n, DH(pk_B, sk_A))

NaCl

$c = Enc(m, n, pk_A, sk_B)$

$m = Dec(c, n, pk_B, sk_A)$

DH + KDF + SK Enc

…what's missing here?

NaCl

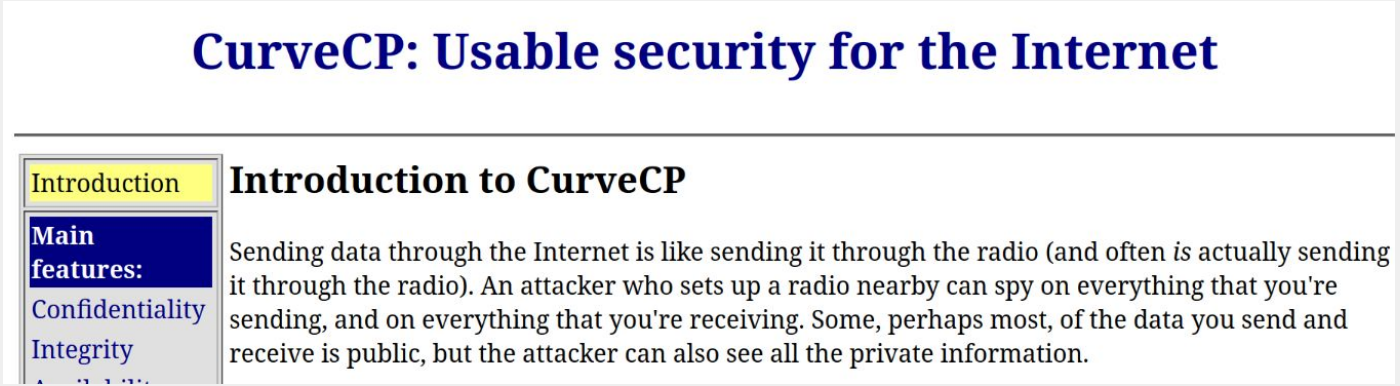
```
c = Enc(m, n, label, pk_A, sk_B)

m = Dec(c, n, label, pk_B, sk_A)
```

DH + KDF + SK Enc

…what's missing here?

NaCl



CurveCP: Usable security for the Internet

Introduction

Main features:

Confidentiality

Integrity

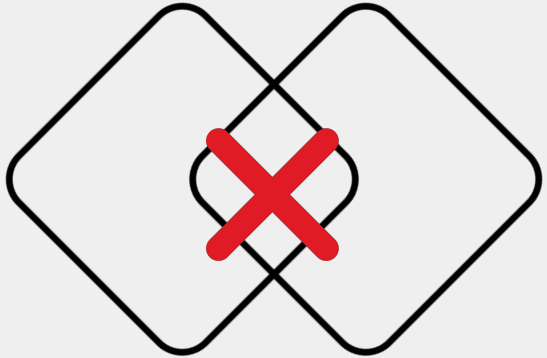
Introduction to CurveCP

Sending data through the Internet is like sending it through the radio (and often *is* actually sending it through the radio). An attacker who sets up a radio nearby can spy on everything that you're sending, and on everything that you're receiving. Some, perhaps most, of the data you send and receive is public, but the attacker can also see all the private information.

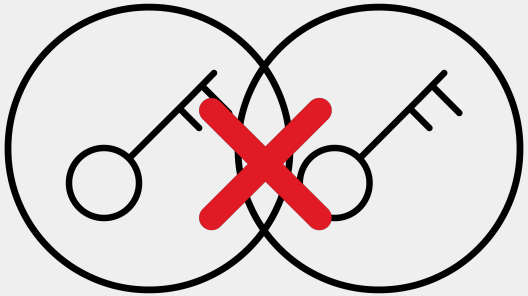
```
Server Hello: "CurveCP-server-M\0\0\0\0\0\0\0\0",
Vouch: "CurveCPV\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0",

Client to server: "CurveCP-client-H\0\0\0\0\0\0\0\0",
Server to client: "CurveCP-server-M\0\0\0\0\0\0\0\0"
```

NaCl

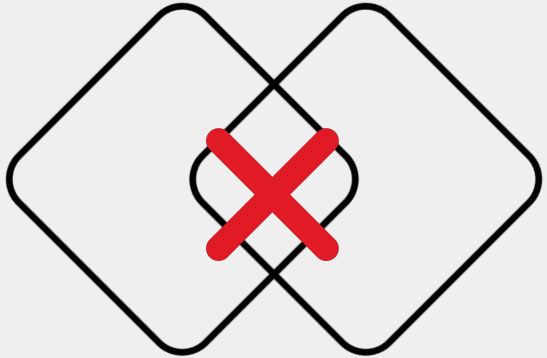


Domain Separation

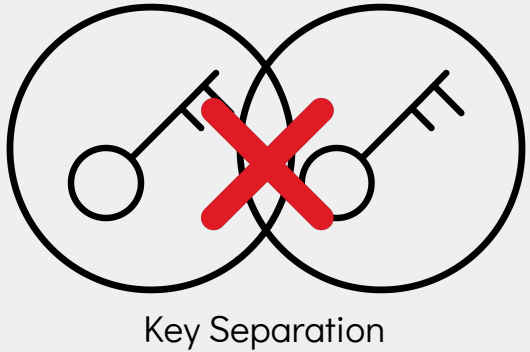


Key Separation

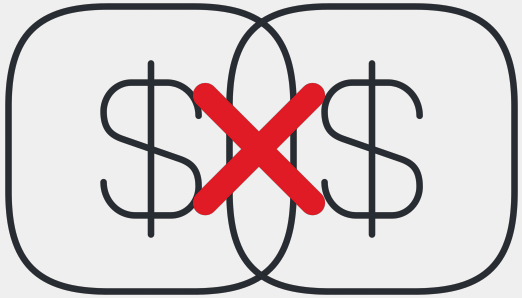
NaCl



Domain Separation



Key Separation



Nonce Separation

Conclusions

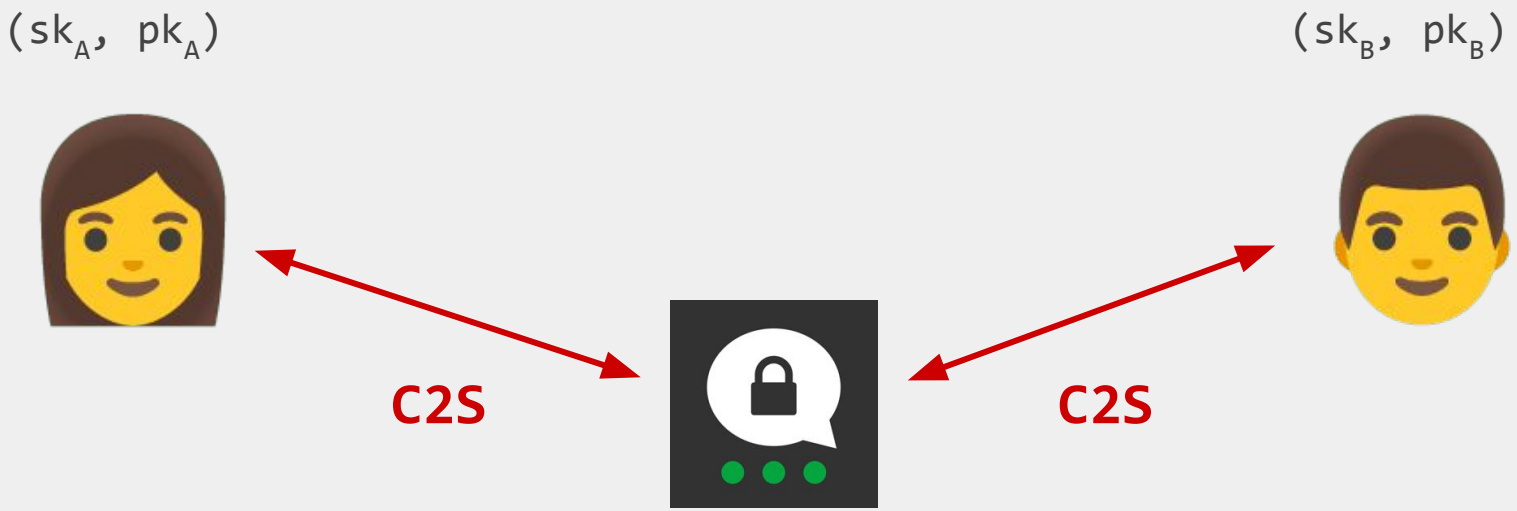
Jacques Gabriel le Fils Huquier (1730 - 1805) - Vue des Enfers

Conclusions

- Missing incentives to keep your crypto up-to-date!
- Library APIs can lead to bad design!
- Backwards compatibility is an obstacle to secure design!

Extra Slides

What do we want? A Secure Channel!

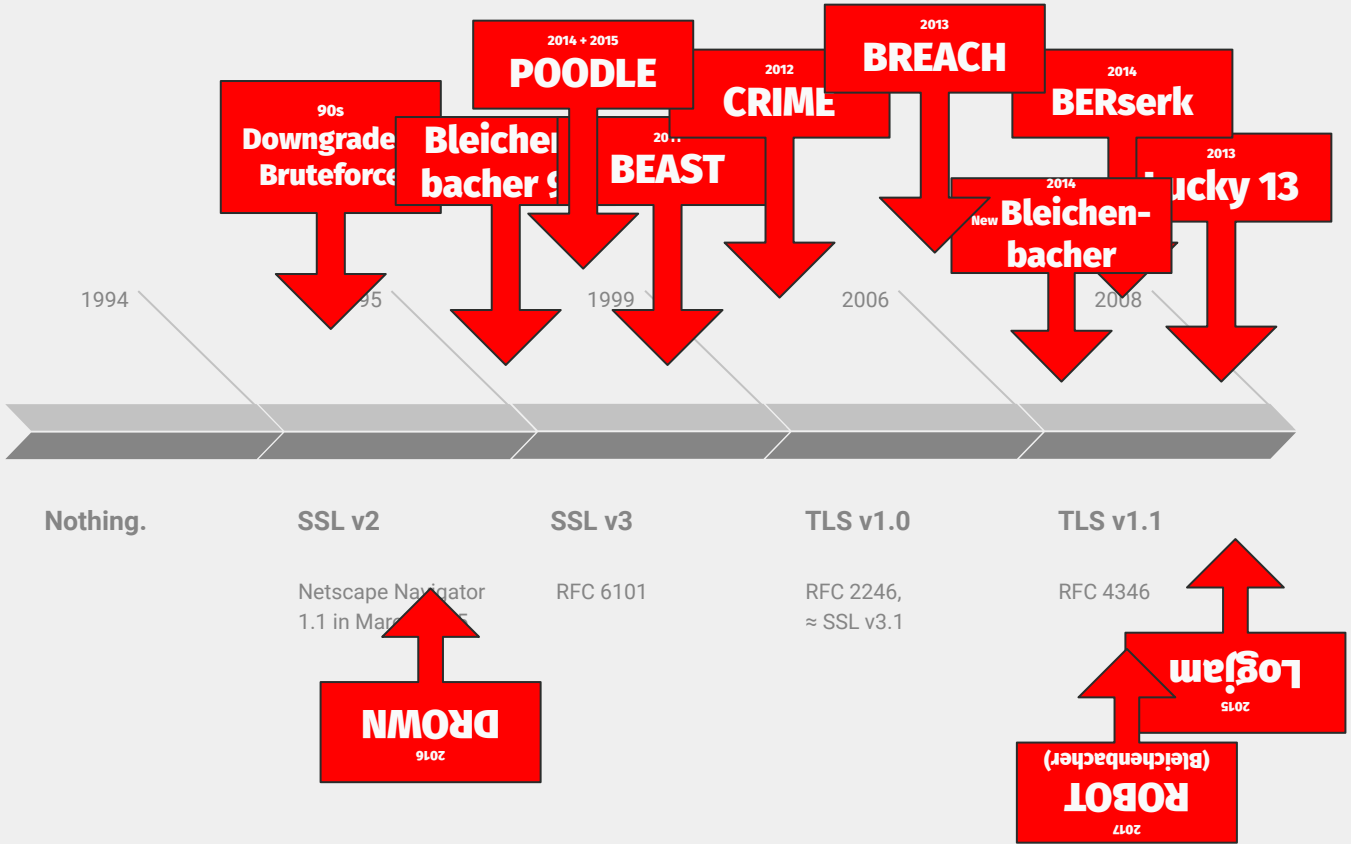


Secure Channels in 2012

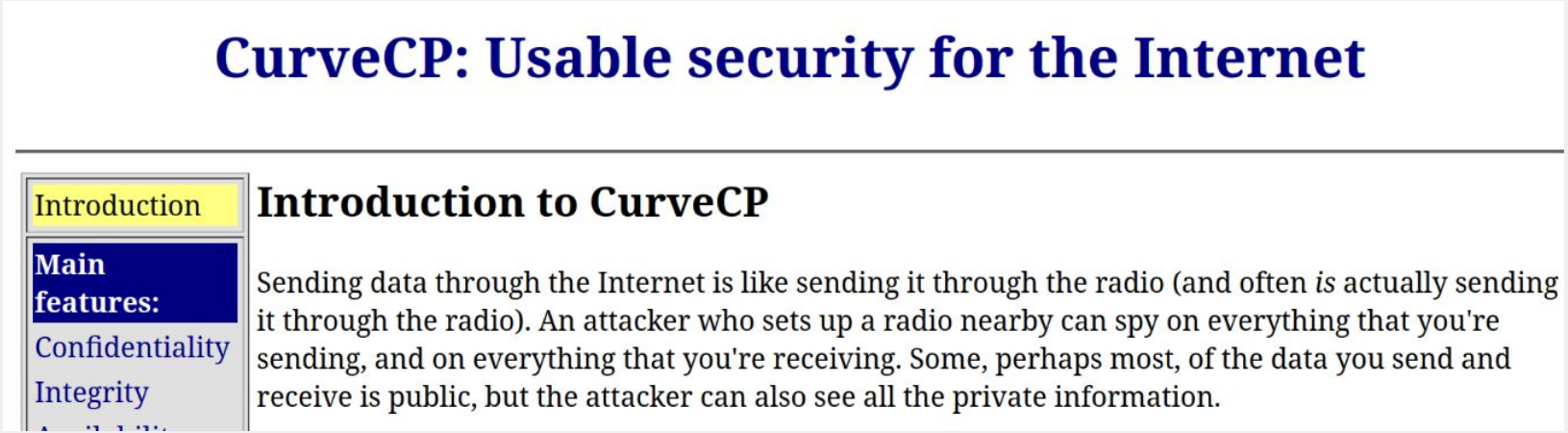
2008: TLS 1.2

2013: Chrome supports TLS 1.2

Secure Channels in 2012



CurveCP



CurveCP: Usable security for the Internet

Introduction

Main features:

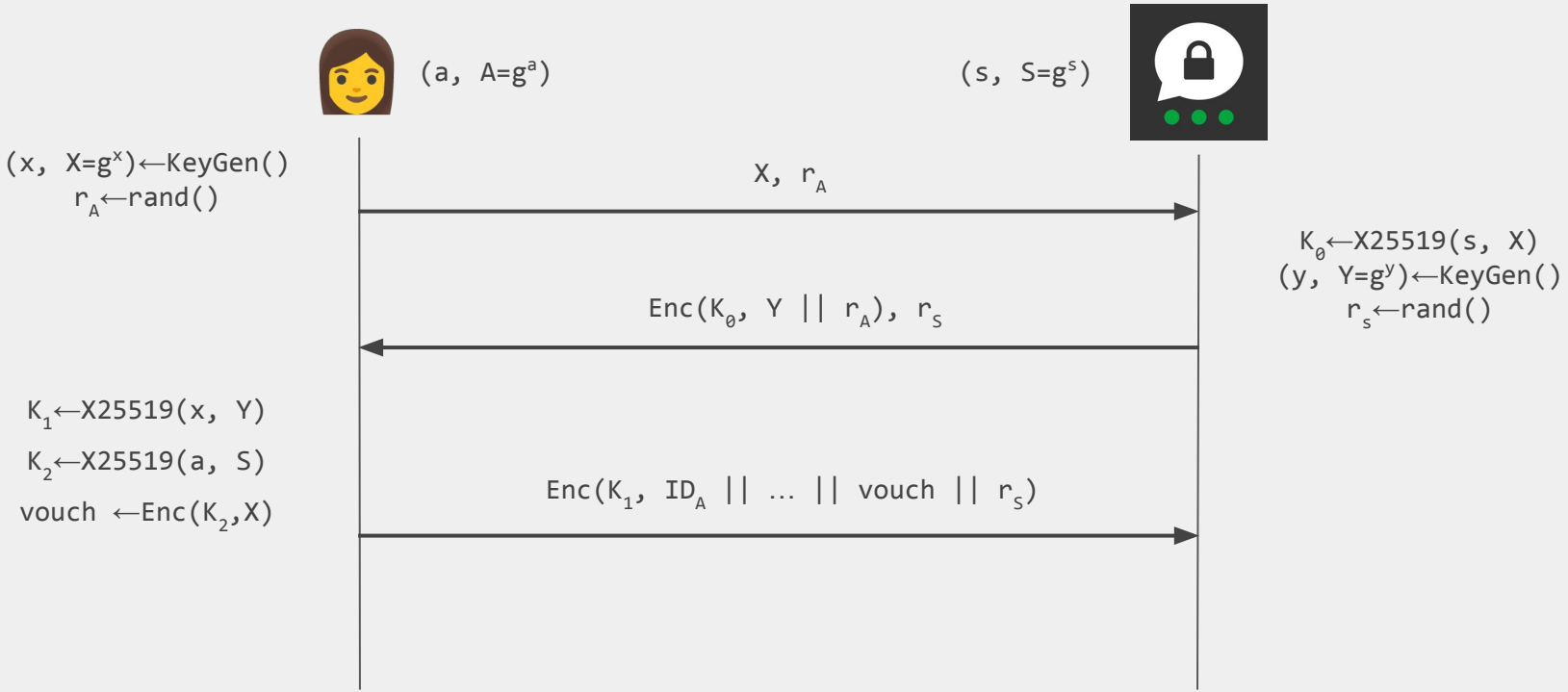
Confidentiality

Integrity

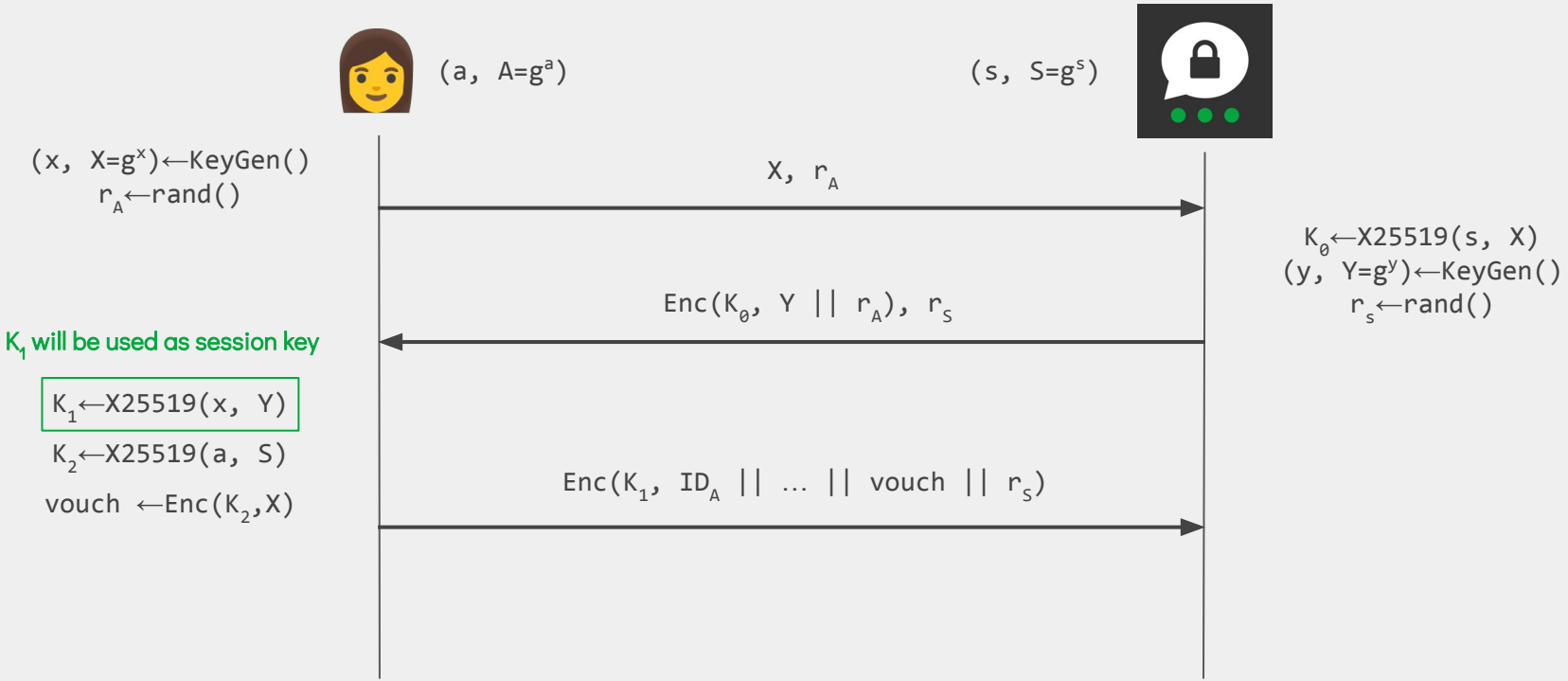
Introduction to CurveCP

Sending data through the Internet is like sending it through the radio (and often *is* actually sending it through the radio). An attacker who sets up a radio nearby can spy on everything that you're sending, and on everything that you're receiving. Some, perhaps most, of the data you send and receive is public, but the attacker can also see all the private information.

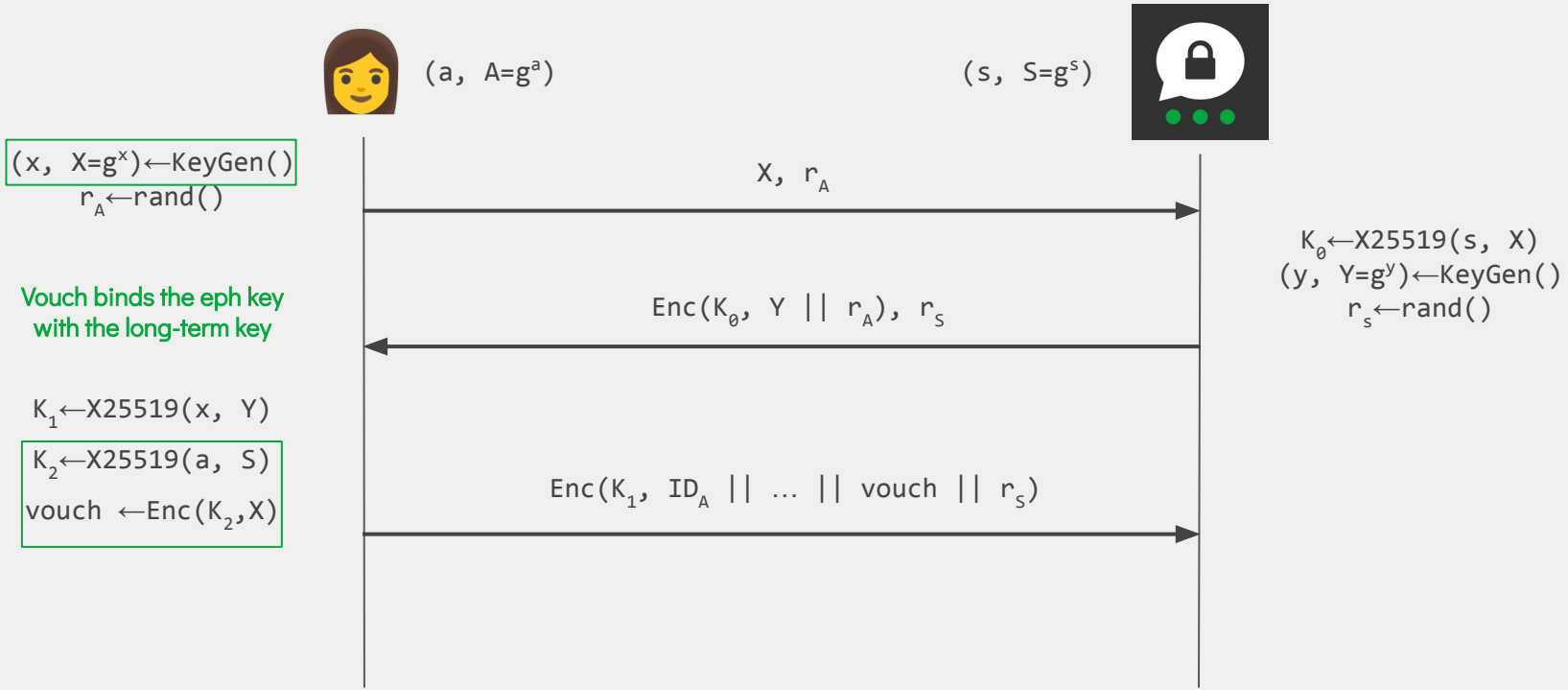
The C2S Protocol: Handshake



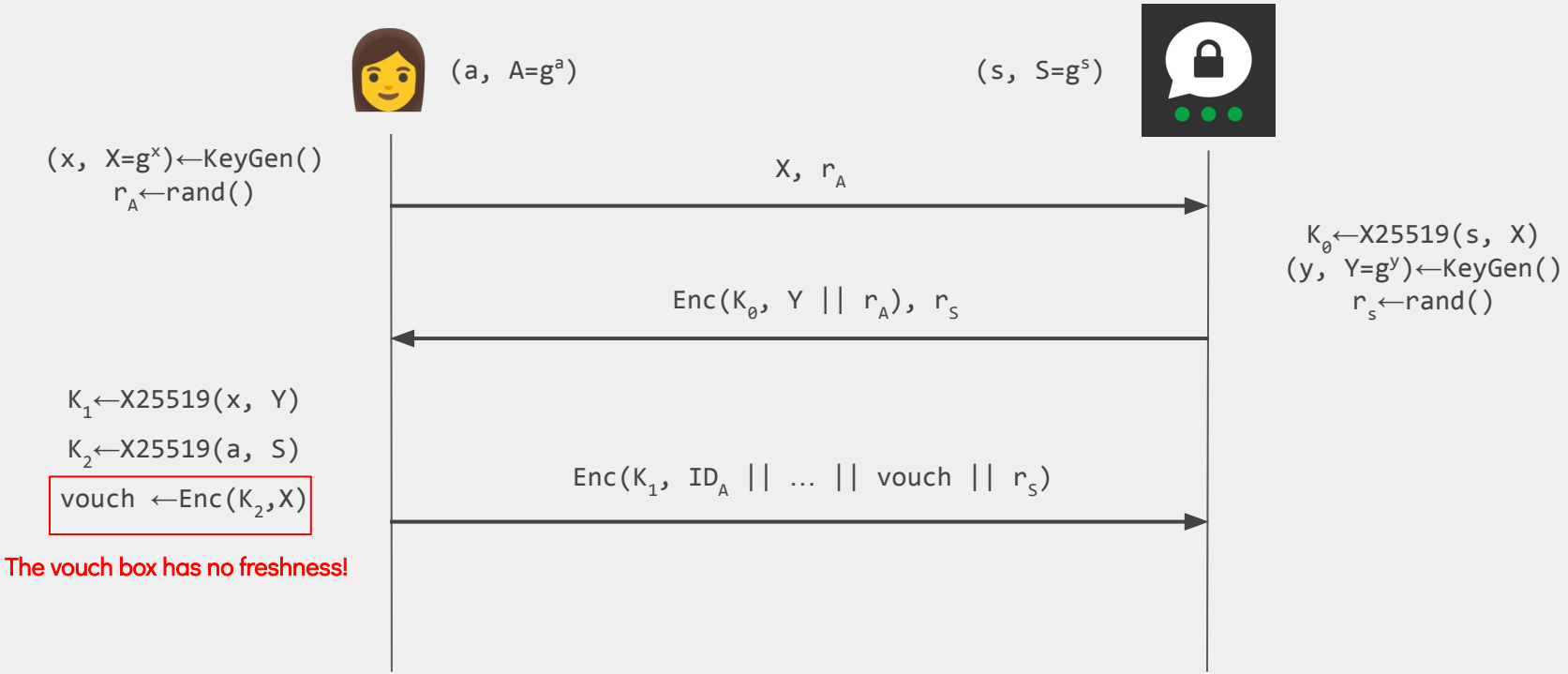
The C2S Protocol: Handshake



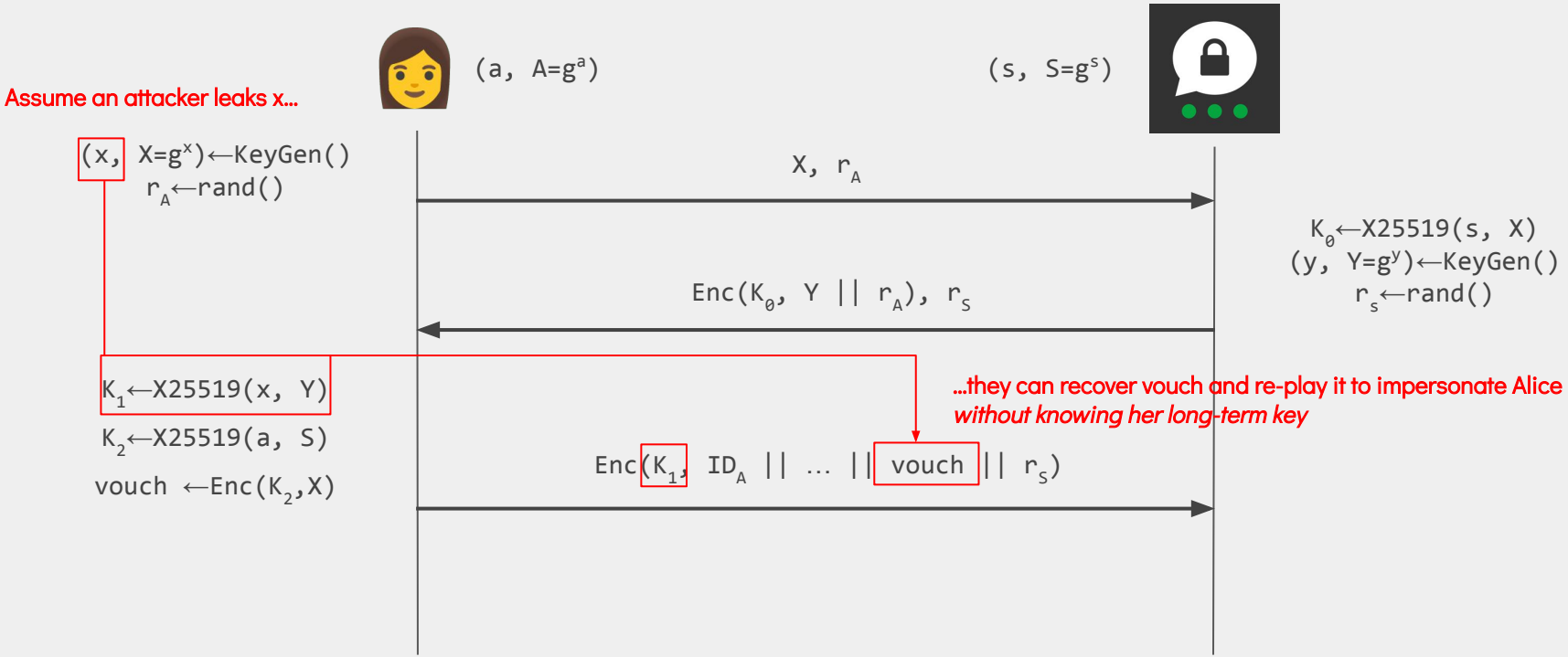
The C2S Protocol: Handshake



The C2S Protocol: Handshake



The C2S Protocol: Handshake



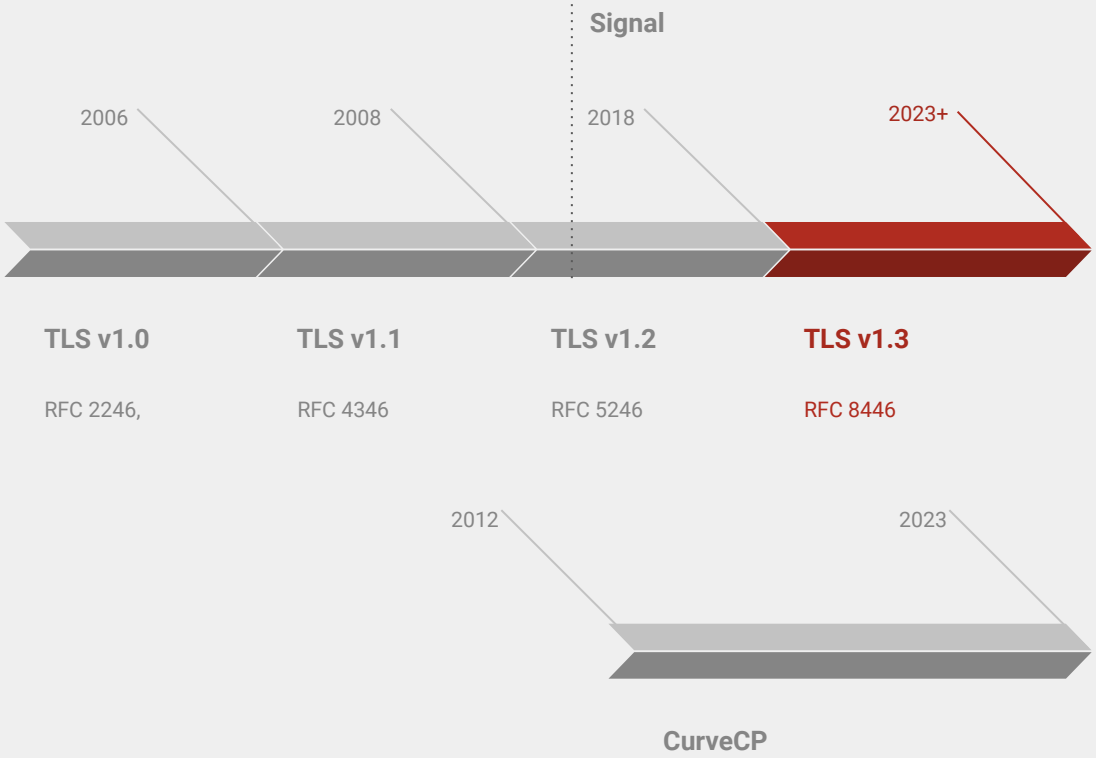
What went wrong?

CurveCP: Usable security for the Internet

Introduction to CurveCP

Sending data through the Internet is like sending it through the radio (and often *is* actually sending it through the radio). An attacker who sets up a radio nearby can spy on everything that you're sending, and on everything that you're receiving. Some, perhaps most, of the data you send and receive is public, but the attacker can also see all the private information.

What went wrong?



No incentives to update!