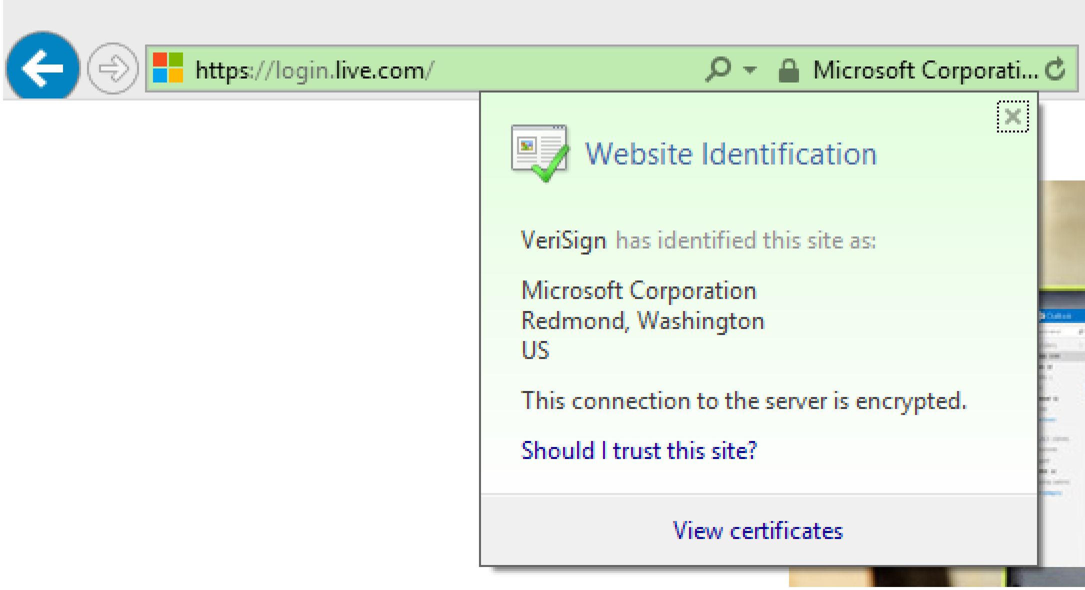
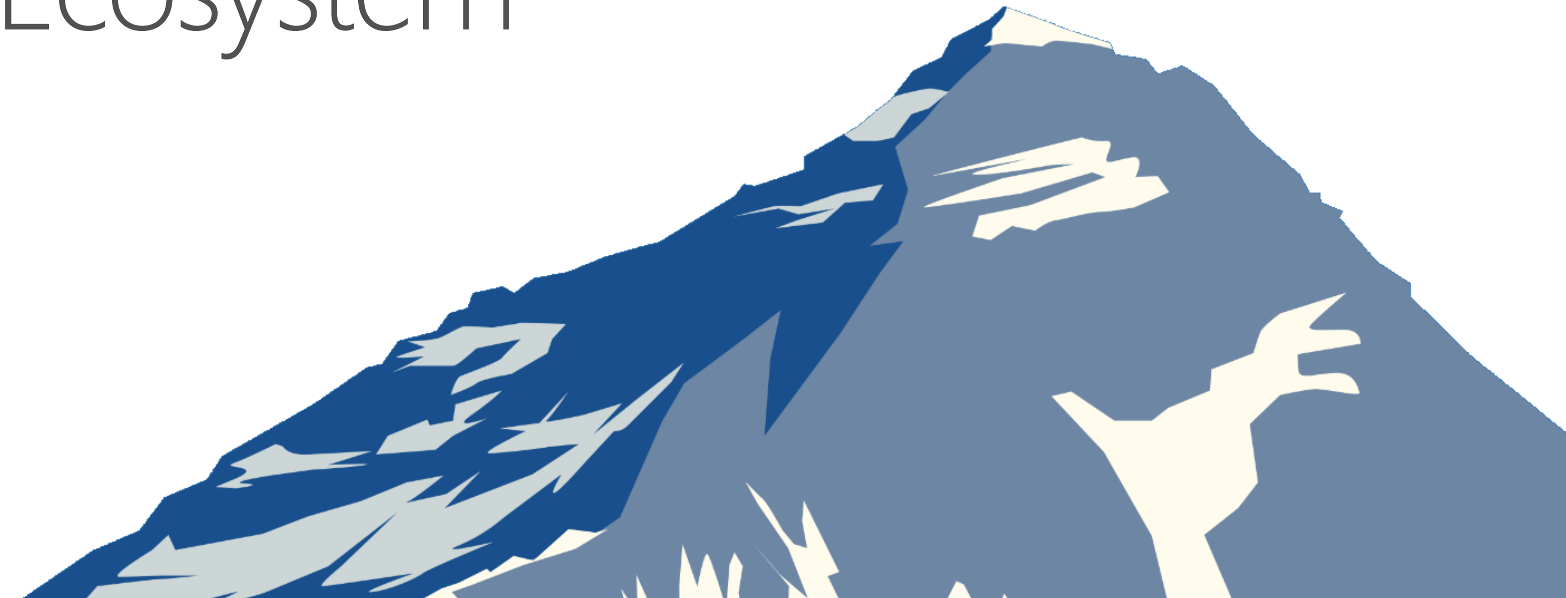
Microsoft Research

Microsoft Research - Inria
JOINT CENTRE



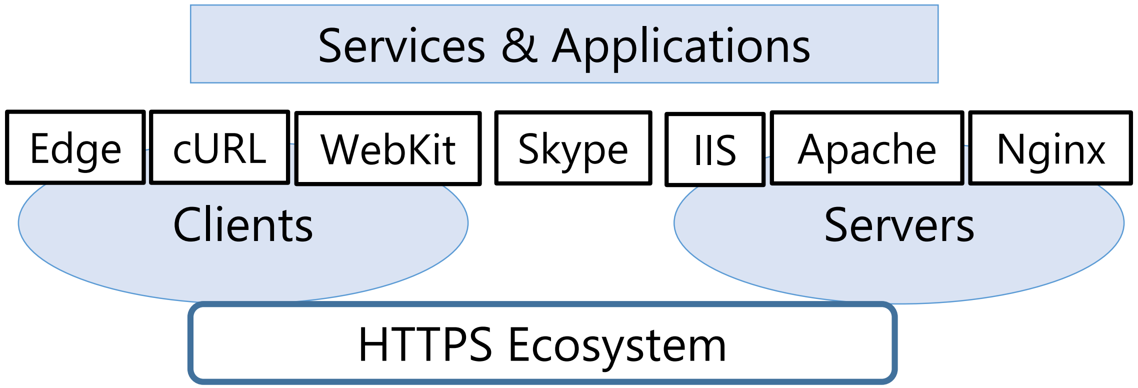
Verified Secure Implementations for the HTTPS Ecosystem

miTLS & Everest*



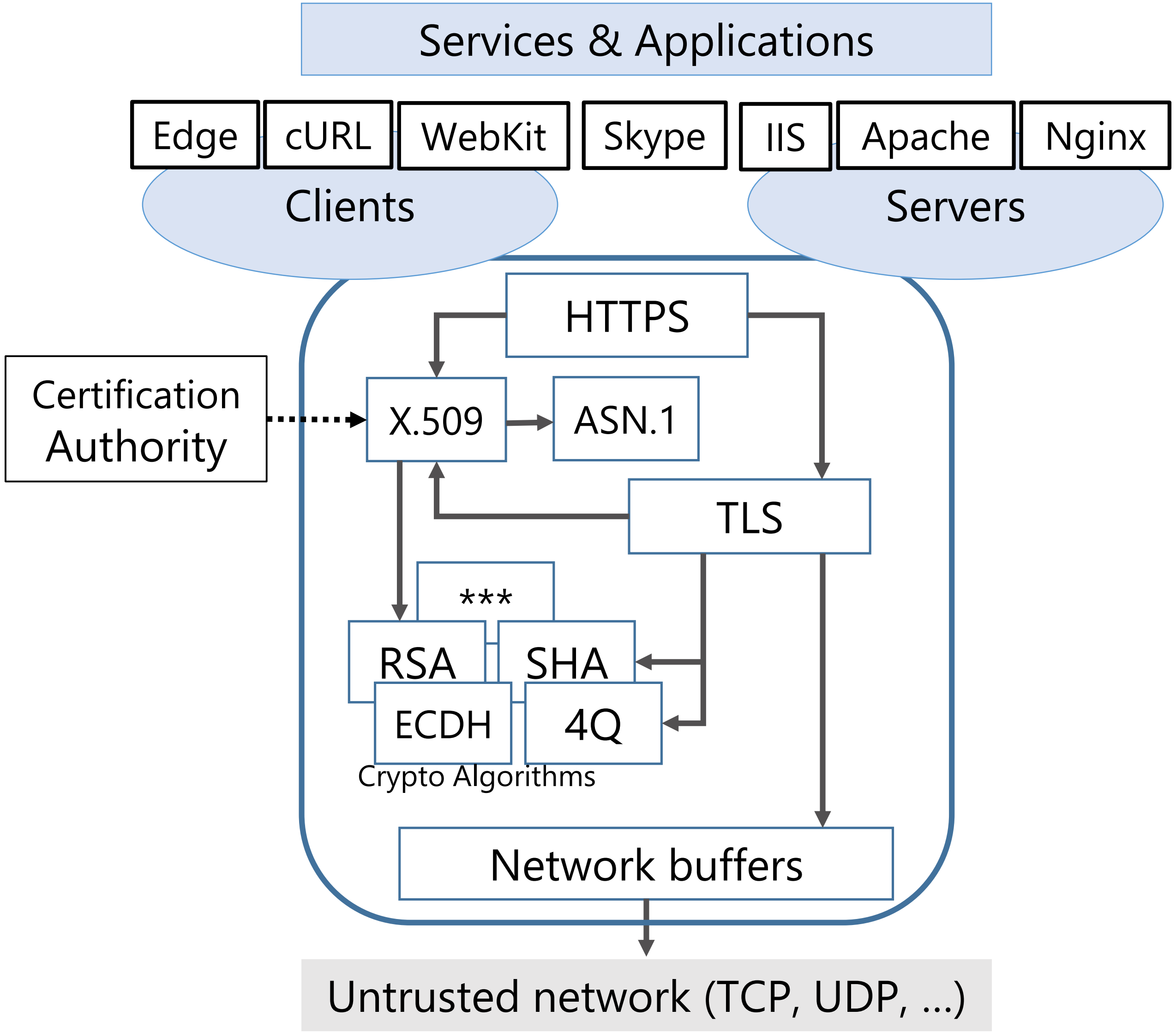
*the Everest VERified End-to-end Secure Transport

The HTTPS Ecosystem is critical



- Most widely deployed security?
  ½ Internet traffic (+40%/year)
- Web, cloud, email, VoIP, 802.1x, VPNs, …

The HTTPS Ecosystem is complex



The HTTPS Ecosystem is broken

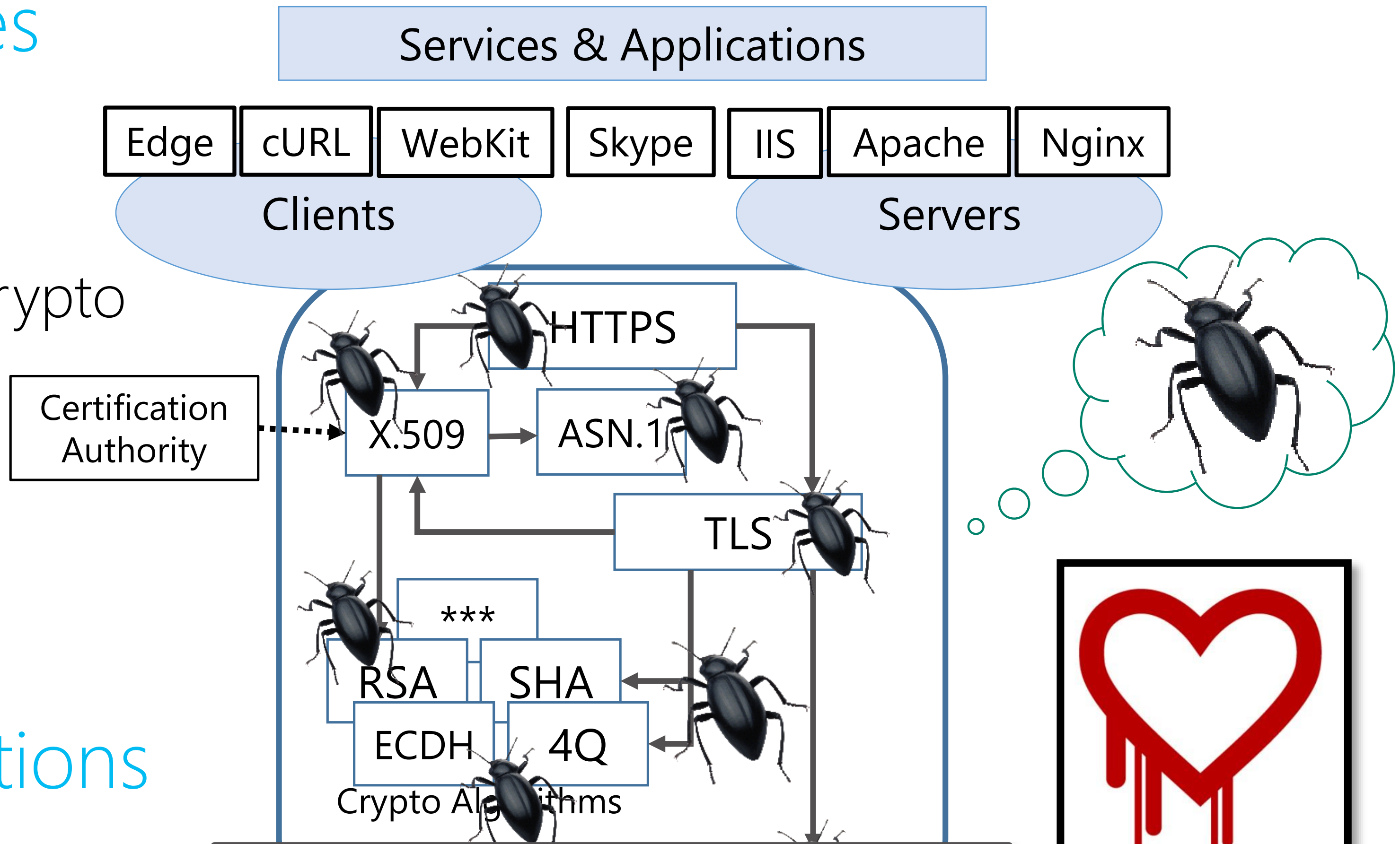
- 20 years of attacks & fixes

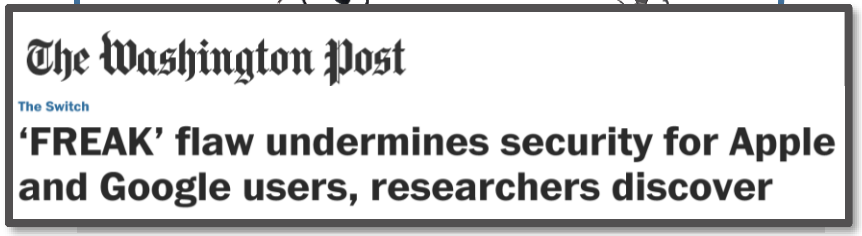
  Buffer overflows
  Incorrect state machines
  Lax certificate parsing
  Weak or poorly implemented crypto
  Side channels

  Informal security goals
  Dangerous APIs
  Flawed standards

- Mainstream implementations

  OpenSSL, SChannel, NSS, …
  Still patched every month!

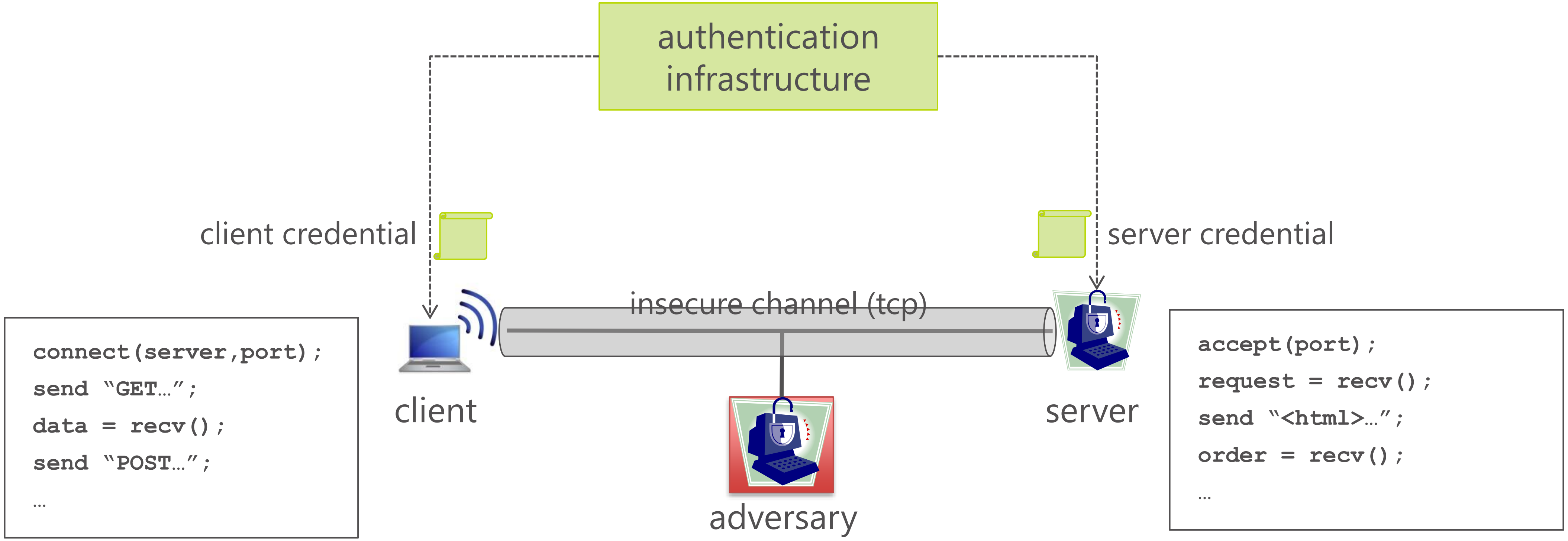



The Washington Post

The Switch

'FREAK' flaw undermines security for Apple and Google users, researchers discover

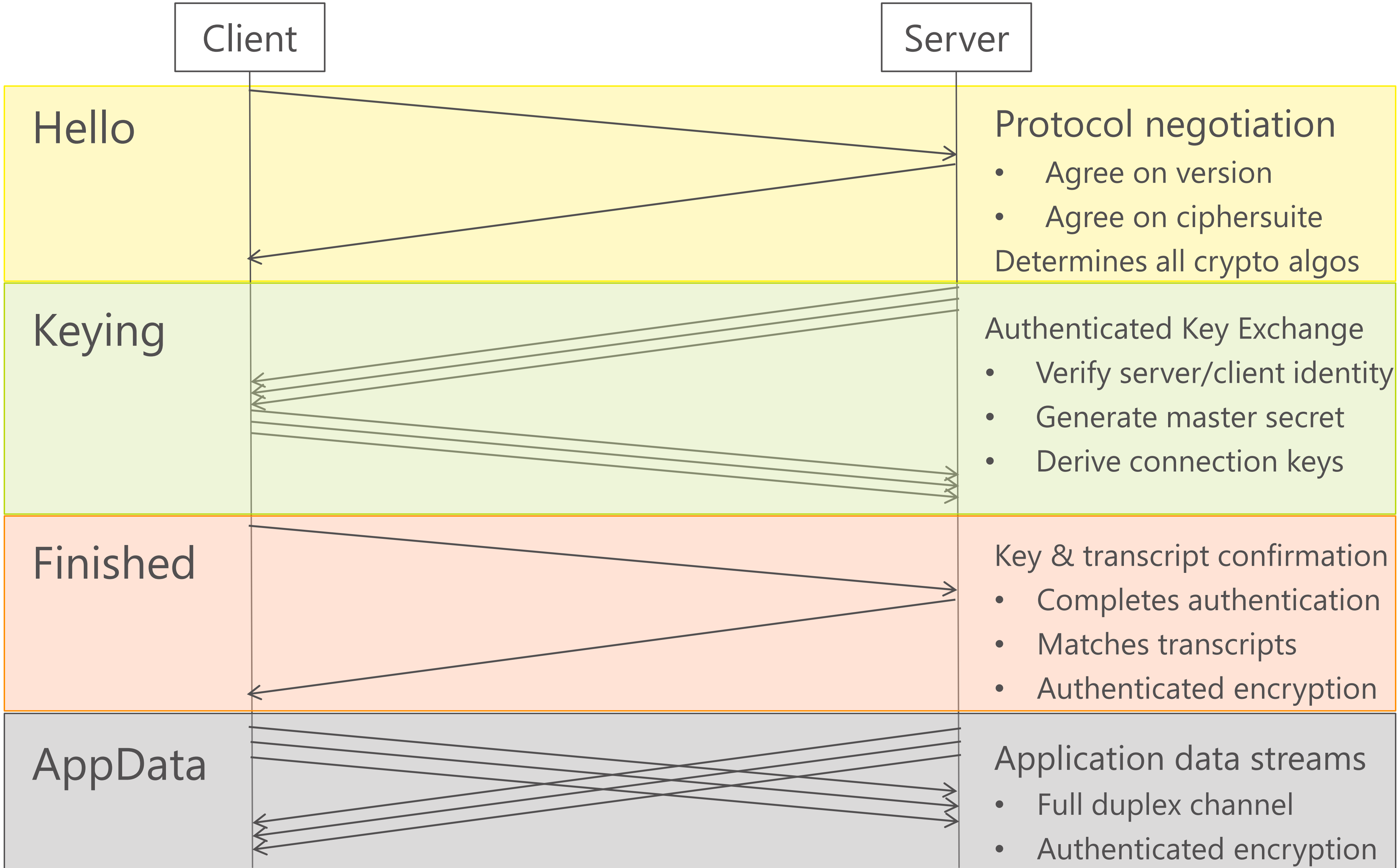
Goal: a secure channel



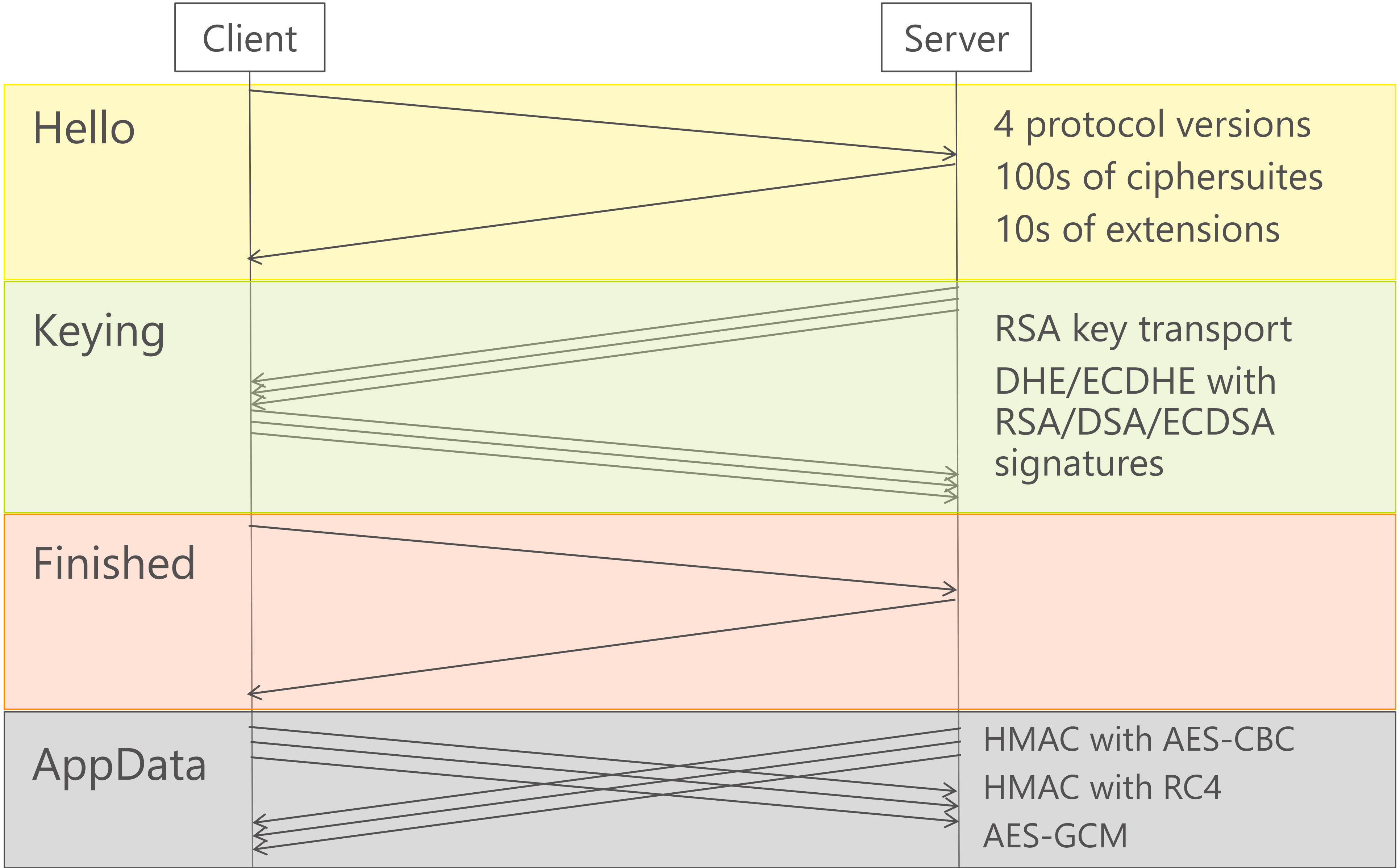
Security Goal: As long as the adversary does not control the long-term credentials of the client and server, it cannot

- Inject forged data into the stream (authenticity)
- Distinguish the data stream from random bytes (confidentiality)

TLS protocol overview



Many configurations (some of them broken)



miTLS (2013—…)
a first verified reference implementation

1. Internet Standard compliance & interoperability
   supporting SSL 3.0—TLS 1.2
2. Verified security:
   we structured our code to enable its modular cryptographic verification, from its main API down to concrete algorithms (RSA, AES,…)
3. Experimental platform:
   for testing corner cases, trying out attacks, analysing extensions and patches, …

Excluding core crypto algorithms

Not fully automated (paper proofs too)

Not production code (poor performance)

GitHub

Search GitHub

miTLS

A verified reference TLS implementation

http://www.mitls.org/

Repositories People 0

Filters ▾ Find a repository…

mitls-fstar

TLS implemented in F*

Updated 17 hours ago

mitls-flex

TLS implemented in f7

Updated 14 days ago

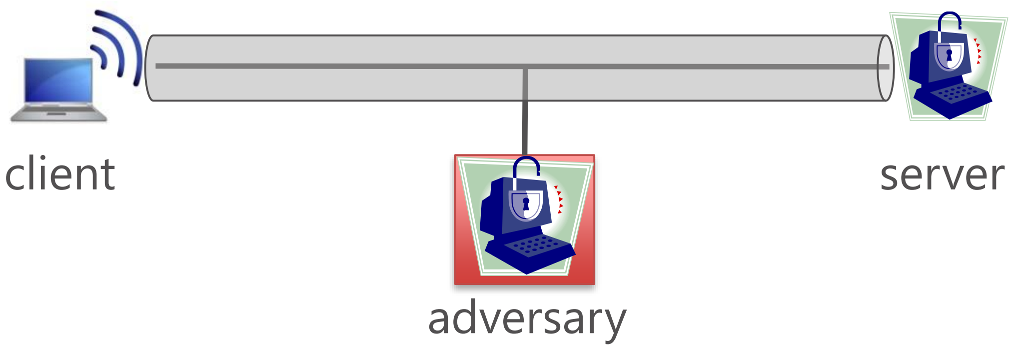
miTLS v0.9 released in Nov'15

https://github.com/mitls

using F* (in progress)
with early support for TLS 1.3

using F# & F7 (stable)
including testing tools

Verified Communications Security?



Application security (API, configuration)
Cryptographic schemes & assumptions
Protocol design
Implementation safety
Information control (leakage, privacy)
Verification tools (F#, F7, F*, Z3, Lean)

(1) data streams
(2) main theorem
(3) state-machine attacks

Modelling Secure Data Streams (1/2)

Type abstraction for integrity & confidentiality

Ideally, TLS passes around data fragments; it cannot forge them, or read their contents.

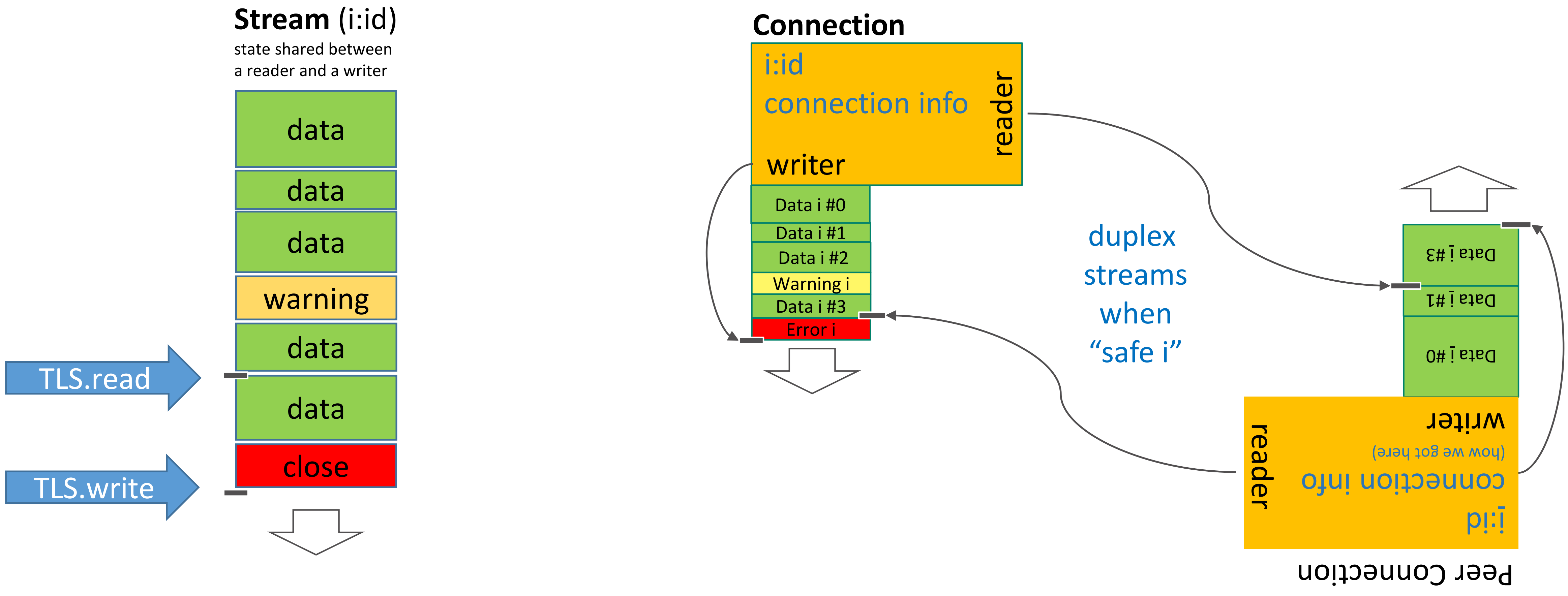
Concretely, this reduces to probabilistic poly-time security assumptions on the underlying cryptographic primitives (e.g. INT-CCA + IND-CPA)

We use **type indexes** to separate between different streams, keep track of their lengths, and control coercions to concrete bytes

```
// F* definition of Application Data

abstract type data (i:id) = bytes

let ghost #(i:id) (d:data i): GTot bytes = d

type fragment (i:id) (rg:range) =
  d:data i {within (ghost d) rg}

val repr: i:id{¬safe i}
  → rg:range
  → d:fragment i rg
  → Tot (b:bytes {b = ghost d})

val make: i:id{¬safe i}
  → rg:range
  → b:bytes{within b rg}
  → Tot (d:fragment i rg {b = ghost d})
```

Modelling Secure Data Streams (2/2)

Stateful invariant for stream authentication

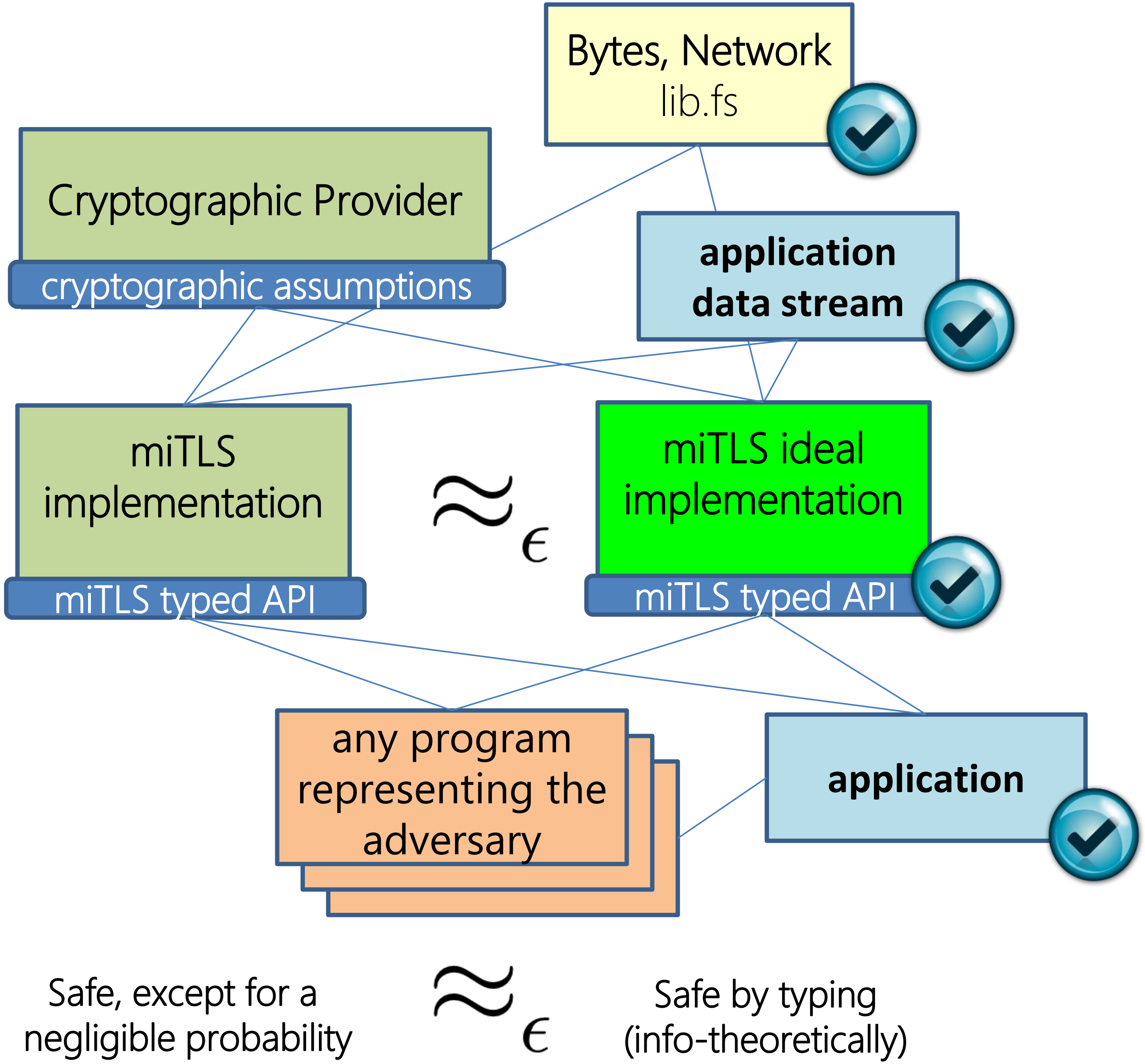


Security Theorem

Main crypto result:

concrete TLS & ideal TLS are **computationally indistinguishable**

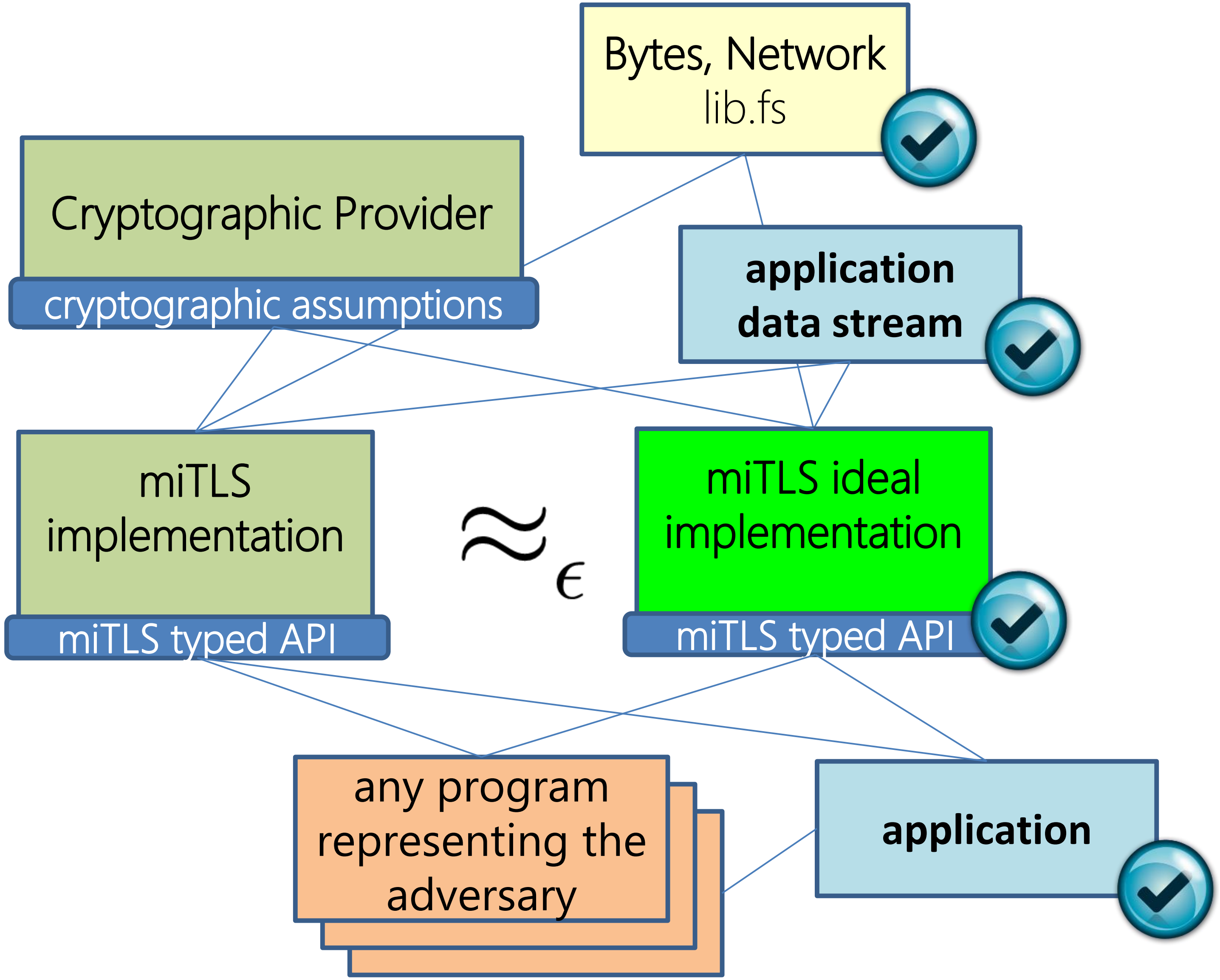
Verification technique: security by typing



Security Theorem

Proof automation

7,000 lines of F#
verified against
3,000 lines of F7
type annotations



The security statement is precise
but complex, roughly the size of the
TLS API and cryptographic assumptions

Scripting Tools & Security Testing

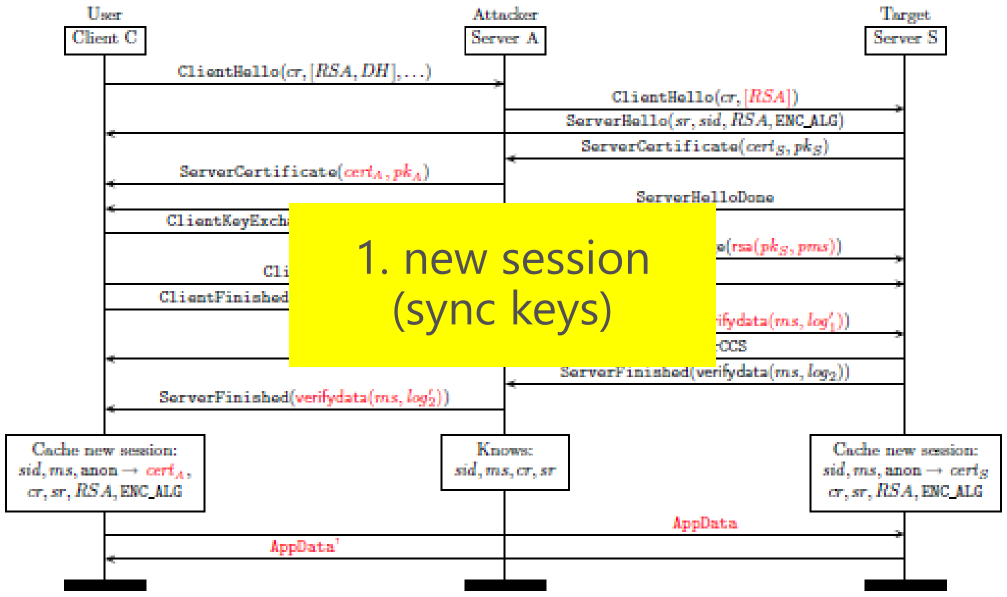
miTLS clean, modular implementation supports rapid prototyping against others

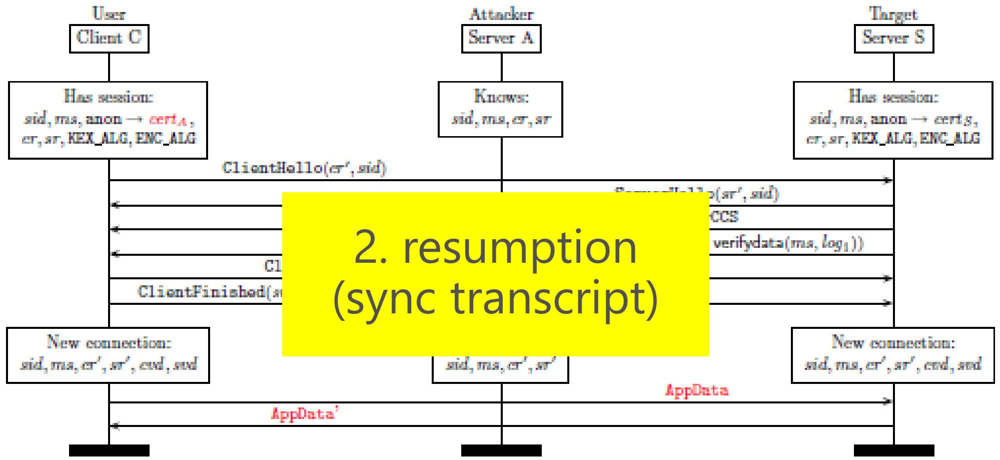
- One line of F# script for each TLS message, with good cryptographic defaults
- Simple setup for “man-in-the-middle” attacks and concurrent connections
- Built-in library of recent vulnerabilities
- Fuzzing on the TLS state machine

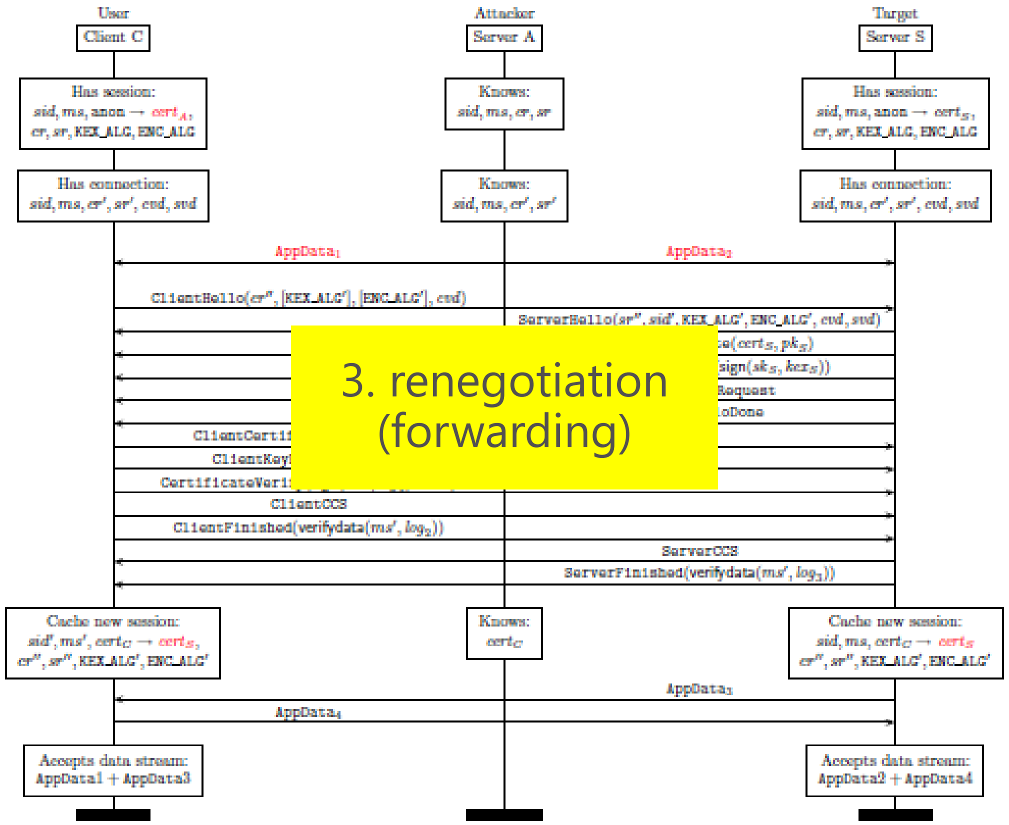
Focus on ease of use (but still for experts)

Triple handshake attack (2014)

flaw in the standard
now patched in TLS







https://www.secure-resumption.com/

Systematically testing the TLS state machine

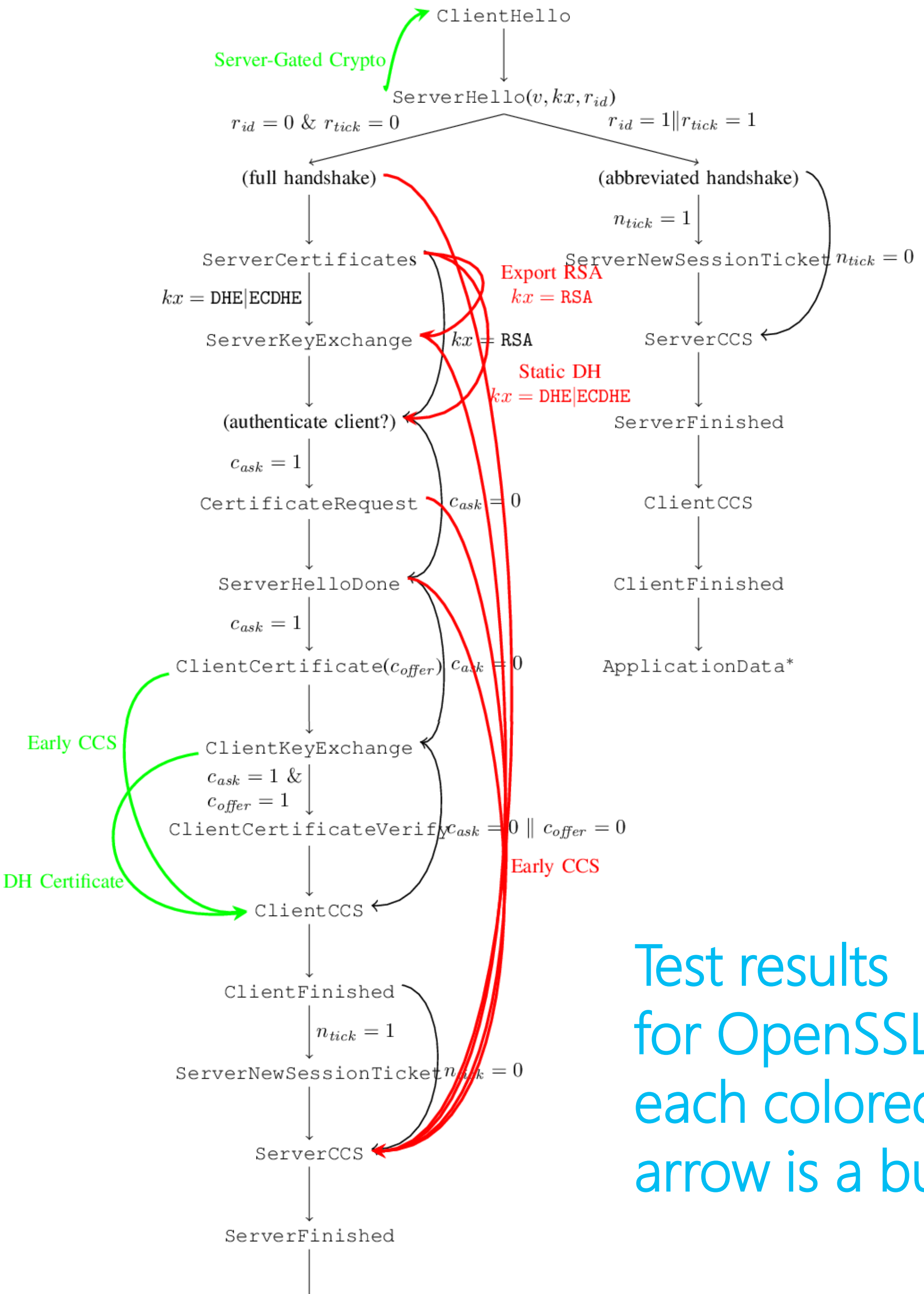
new attacks against all mainstream implementations

TLS offers many ciphersuites, optional messages, extensions… sharing the same state machine.

miTLS provides a verified TLS state machine.

We systematically generate and test deviant traces against other implementation (skipping, inserting, reordering valid messages)

We found many many exploitable bugs



Test results for OpenSSL: each colored arrow is a bug

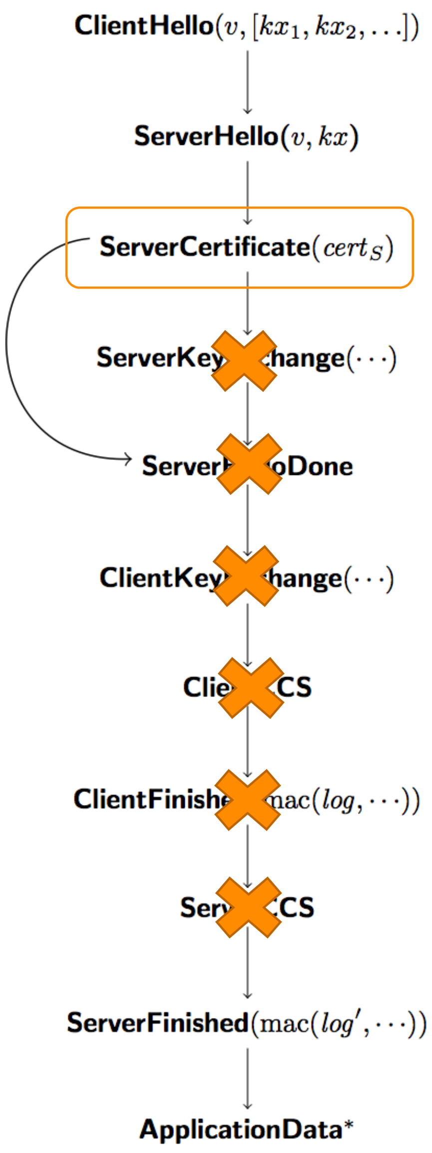
Systematically testing the TLS state machine

new attacks against all mainstream implementations

TLS offers many ciphersuites, optional messages, extensions… sharing the same state machine.

miTLS provides a verified TLS state machine.

We systematically generate and test deviant traces against other implementation (skipping, inserting, reordering valid messages)



An attack against TLS Java Library (open for 10 years)

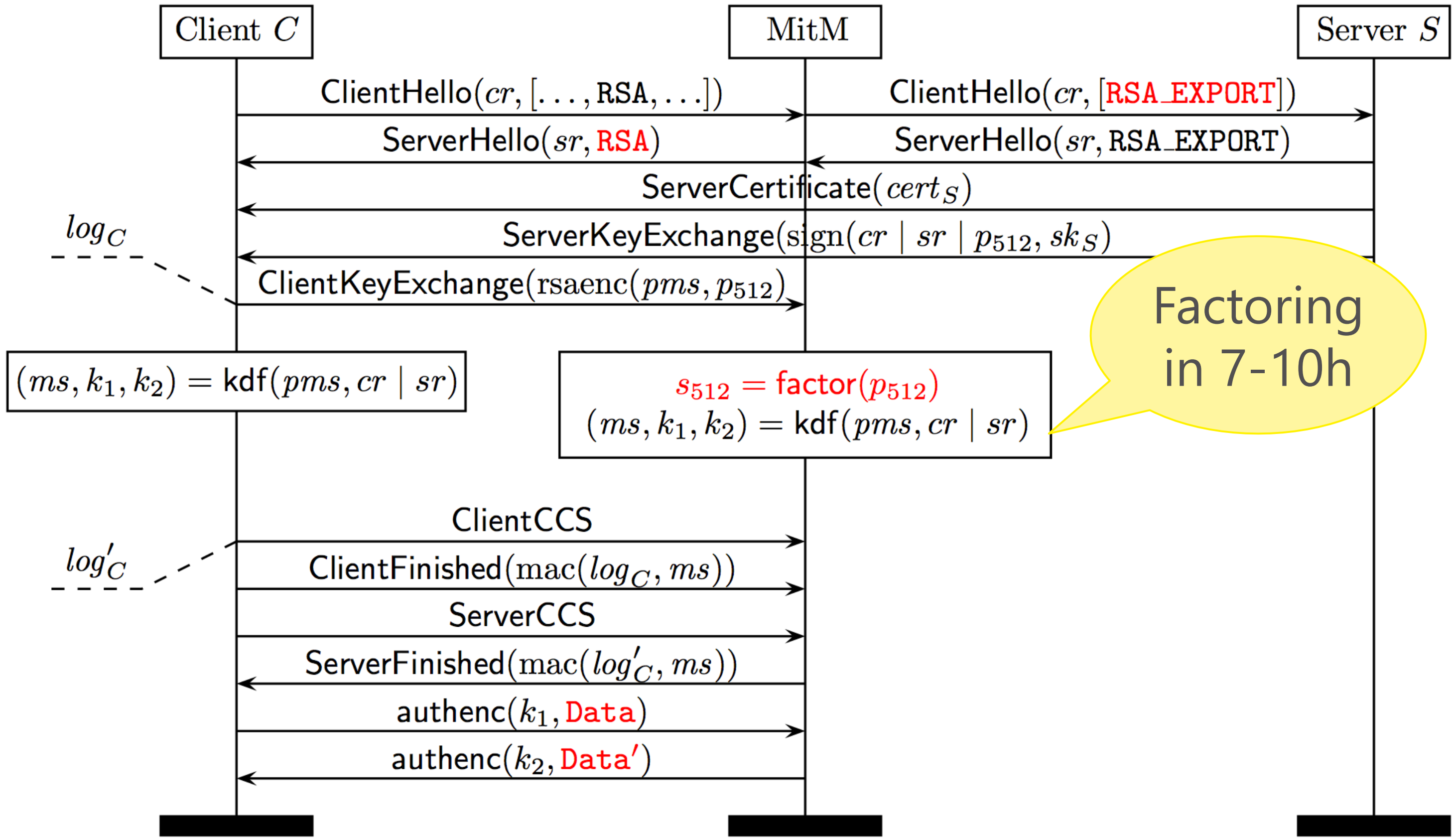
We skip 6 messages

JSSE's client assumes the key exchange is finished, uses uninitialized 0x000000… as session key!

FREAK: downgrade to RSA_EXPORT (2015)

Man-in-the-middle attack against:

- servers that support RSA_EXPORT (512bit keys obsoleted in 2000) from 40% to 8.5%
- clients that accept ServerKeyExchange in RSA (state machine bug) almost all browsers have been patched



Similar attack, different crypto:

LOGJAM (2015) downgrade to weak groups

FREAK in the news

BBC | Menu | Sections

NEWS

Technology

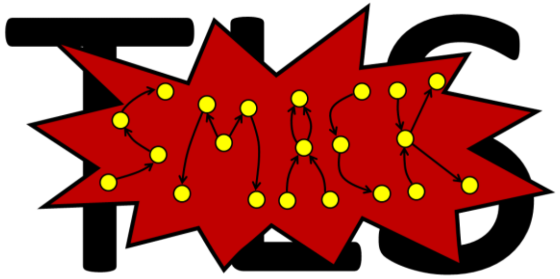
Millions at risk from 'Freak' encryption bug

6 March 2015 | Technology

The Washington Post

The Switch

'FREAK' flaw undermines security for Apple and Google users, researchers discover



The New York Times

Apple, Android Browsers Vulnerable to 'FREAK Attack'

By THE ASSOCIATED PRESS MARCH 3, 2015, 9:06 P.M. E.S.T.

theguardian
Winner of the Pulitzer prize 2014

home › tech | UK | world | politics | sport | football | opinion | culture | business | all

Technology

Apple and Google 'FREAK attack' leaves millions of users vulnerable to hackers

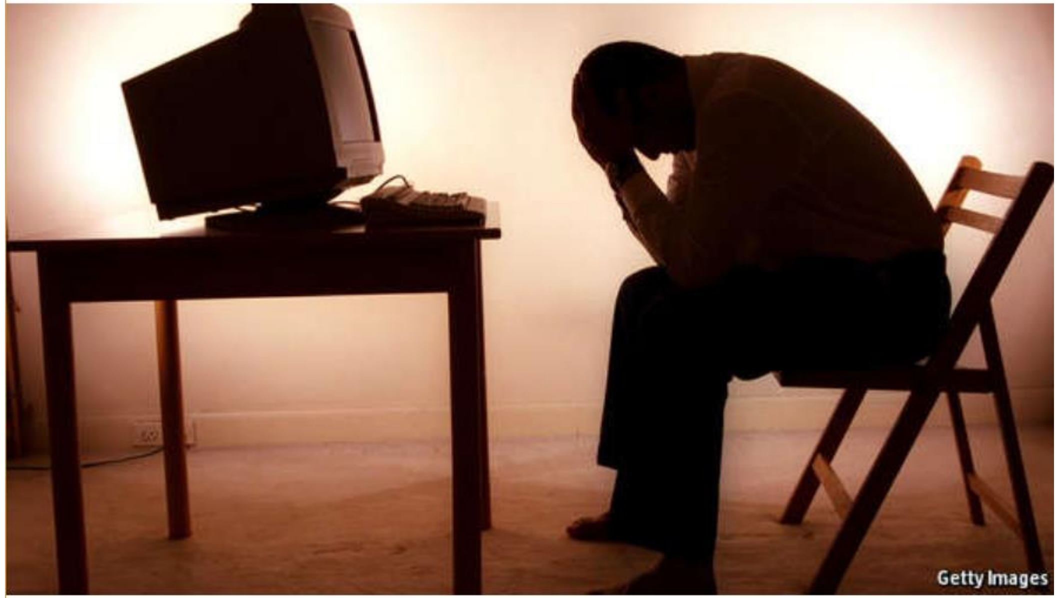
The Economist | World politics | Business & finance | Economics | Science & technology | Cultu

Computer security

The law and unintended consequences

The perils of deliberately sabotaging security

Mar 7th 2015 | From the print edition | Timekeeper | Like 1.1k | Tweet 134



COMPUTERS are notoriously insecure. Usually, this is by accident rather than design. Modern operating systems contain millions of lines of code, with millions more in the applications that do the things people want done. Human brains are simply too puny to build something so complicated without making mistakes.

On March 3rd, though, a group of researchers at Microsoft, an American computer company, Imdea, a Spanish research institute, and the National Institute for Research in Computer Science and Automation, in France, discovered something slightly different. They found a serious flaw in cryptography designed to guard private data such as e-mails,

LOGJAM
in the news

engadget UK REVIEWS FEATURES GUIDES VIDEOS GALLERIES PUBLIC ACCESS GAMING EDITI

MUST READ: Pandora buys Rdio for $75 million

'Logjam' browser vulnerability fix will block thousands of websites

by Steve Dent | @stevetdent | May 20th 2015 At 9:45am

ars technica

HTTPS-crippling attack threatens tens of thousands of Web and mail servers

Diffie-Hellman downgrade weakness allows attackers to intercept encrypted data.

by **Dan Goodin** - May 20, 2015 6:54am BST

ELECTRONIC FRONTIER FOUNDATION
DEFENDING YOUR RIGHTS IN THE DIGITAL WORLD

MAY 27, 2015 | BY JOSEPH BONNEAU

Logjam, Part 1: Why the Internet is Broken Again (an Explainer)

THE WALL STREET JOURNAL.

TECH

New Computer Bug Exposes Broad Security Flaws

Fix for LogJam bug could make more than 20,000 websites unreachable

By JENNIFER VALENTINO-DEVRIES

May 19, 2015 7:02 p.m. ET

black hat

News About Nominations Winners Archive Site search

Pwnie for Most Innovative Research

Awarded to the person who published the most interesti
presentation, tool or even a mailing list post.

- Imperfect Forward Secrecy: How Diffie-He
  Credit: David Adrian et al.

  This paper introduces the Logjam attack, a vulne
  downgrade TLS connections to 512-bit export-gra

We found & fixed flaws in legacy implementations of TLS…
probably many others still in there. Can we be more constructive?

Can we make the next TLS better?

- We are trying to model, implement,
  and improve the new draft standard (TLS 1.3)
- Deployment will take years—are TLS 1.2 and TLS 1.3 jointly secure?

Can we deploy verified code in the TLS/HTTPS ecosystem?

- Despite great technical achievements,
  formally verified software is seldom deployed and used
- TLS is small & critical, can be exemplary case
  for **verified deployed software**

TLS 1.3: a new hope

Much discussions

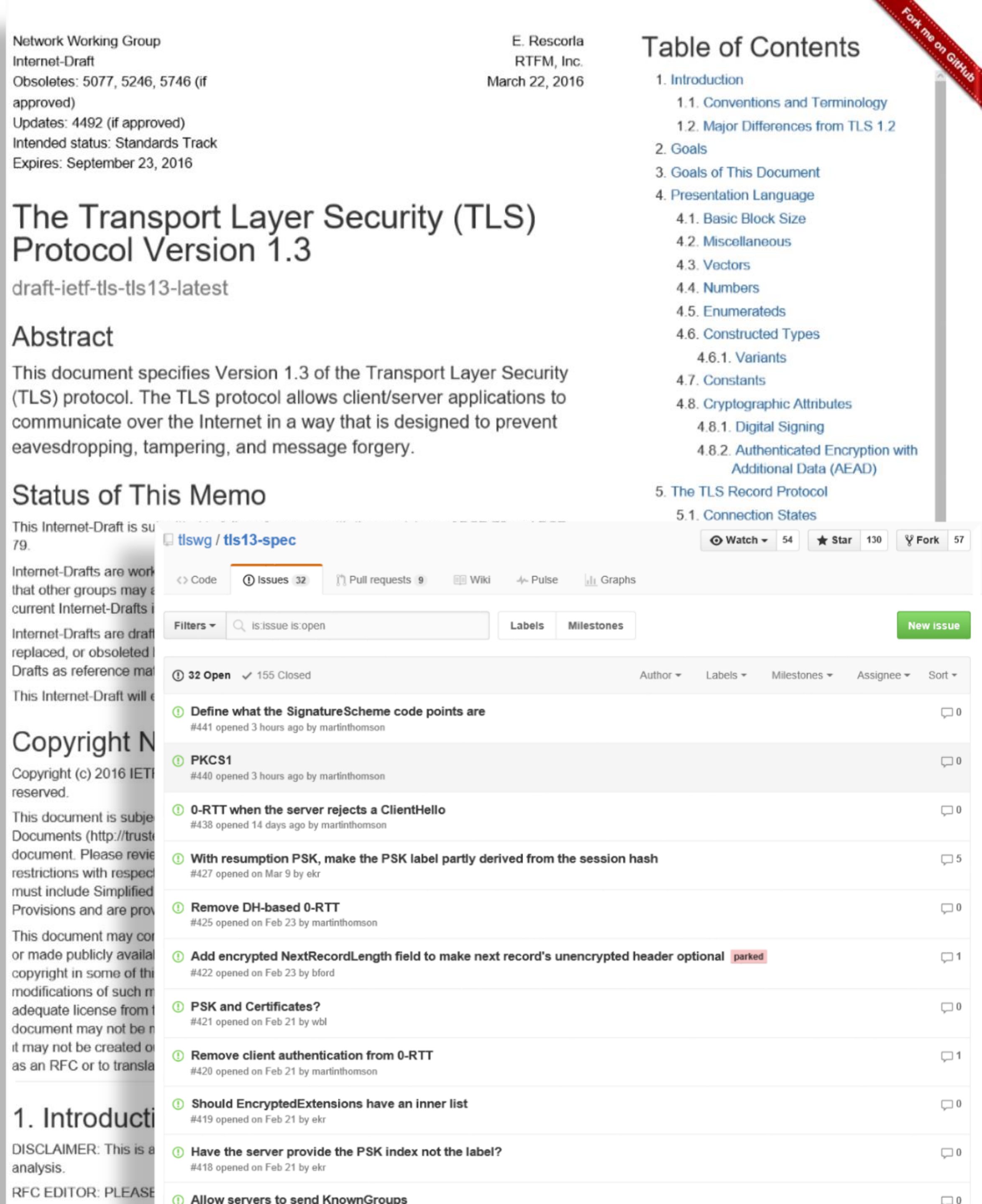
IETF, Google, Mozilla, Microsoft, CDNs, cryptographers, network engineers, …

Much improvements

- Modern design
- Fewer roundtrips
- Stronger security

New implementations required for all

- Be first & verified too!
- Find & fix flaws before it's too late


Network Working Group
Internet-Draft
Obsoletes: 5077, 5246, 5746 (if approved)
Updates: 4492 (if approved)
Intended status: Standards Track
Expires: September 23, 2016

E. Rescorla
RTFM, Inc.
March 22, 2016

The Transport Layer Security (TLS) Protocol Version 1.3
draft-ietf-tls-tls13-latest

Abstract

This document specifies Version 1.3 of the Transport Layer Security (TLS) protocol. The TLS protocol allows client/server applications to communicate over the Internet in a way that is designed to prevent eavesdropping, tampering, and message forgery.

Status of This Memo

Copyright N

1. Introducti

Table of Contents

TLS 1.3: status

IETF TLS WG95
(April'16)

- 13th draft discussed

  Adopting several of our proposals: extended session hashes, downgrade resilience, pre-shared-key 0RTT, session ticket format, simplified key schedule...

- Finalized in 6 months?

Network Working Group
Internet-Draft
Obsoletes: 5077, 5246, 5746 (if approved)
Updates: 4492 (if approved)
Intended status: Standards Track
Expires: September 23, 2016

E. Rescorla
RTFM, Inc.
March 22, 2016

The Transport Layer Security (TLS) Protocol Version 1.3

Implementation Status

Name	Language	ECDHE	DHE	PSK	0-RTT
NSS	C	Yes	No	Yes	Yes*
Mint	Go	Yes	Yes	Yes	Yes
nqsb	OCaml	No	Yes	Yes	No
ProtoTLS	JavaScript	Yes	Yes	Yes	Yes
miTLS	F*	Yes	Yes	Yes	???

- NSS interops with Mint and ProtoTLS
  - NSS 0-RTT in unintegrated branch
- ProtoTLS interops with nqsb
- Other combinations untested

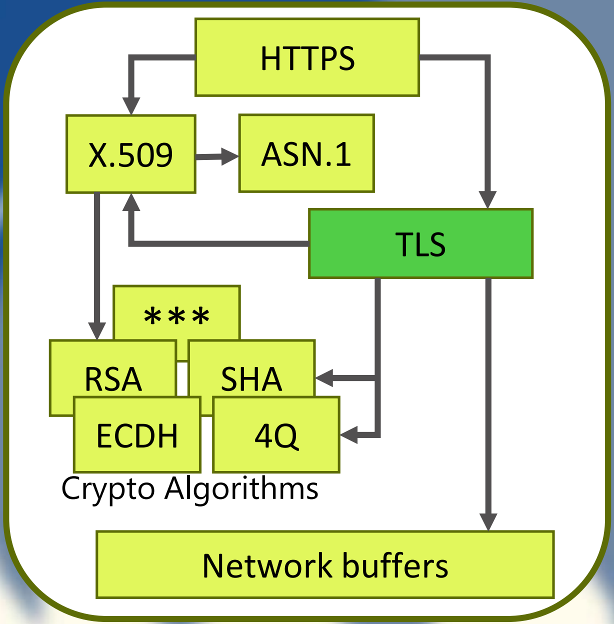
IETF 95

Downgrade Resilience in Key-Exchange Protocols

Karthikeyan Bhargavan, Christina Brzuska, Cédric Fournet, Matthew Green, Markulf Kohlweiss and Santiago Zanella-Béguelin

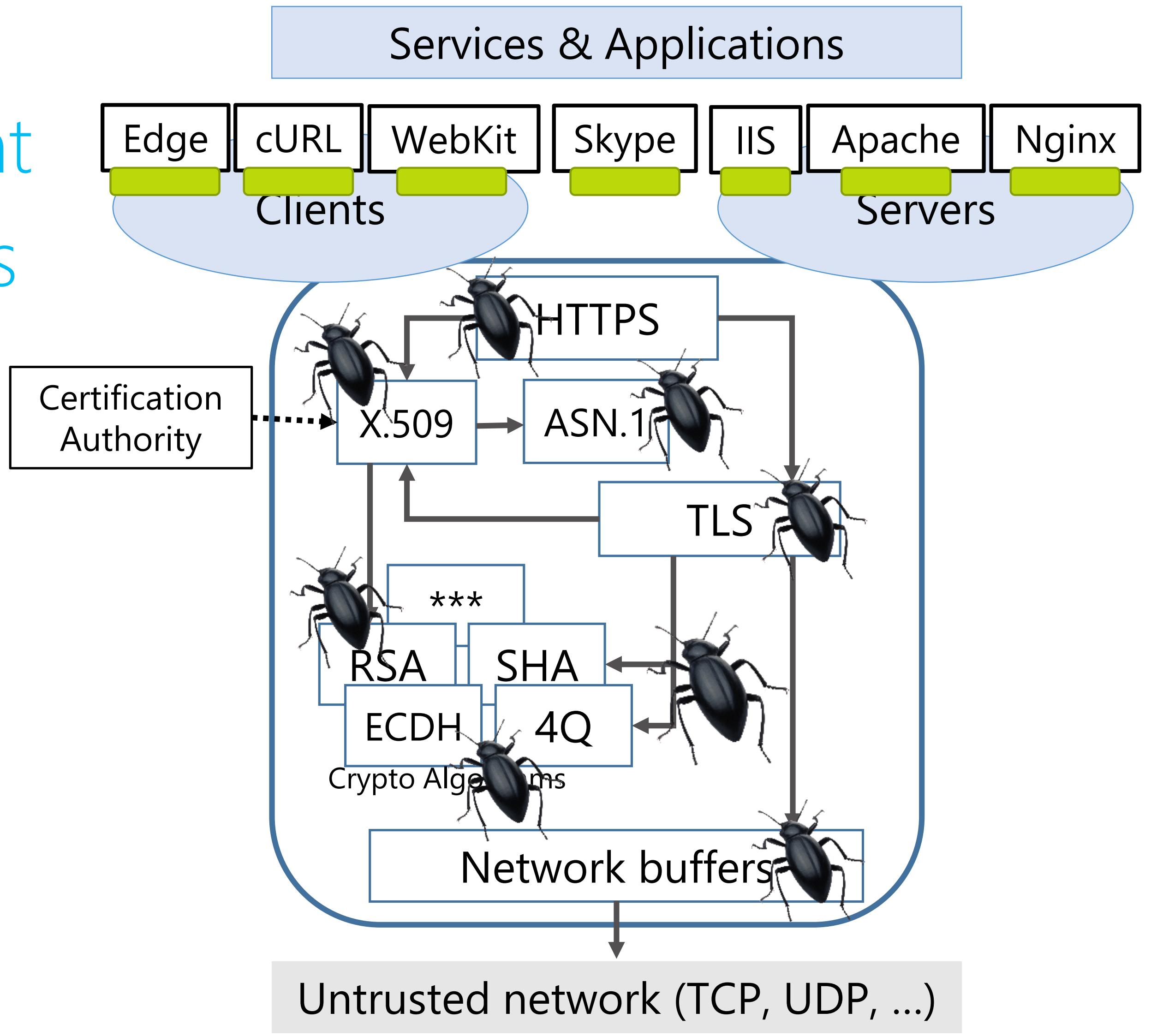
To appear at IEEE S&P

Everest (2016—2021): Verified Drop-In Replacements for the HTTPS ecosystem

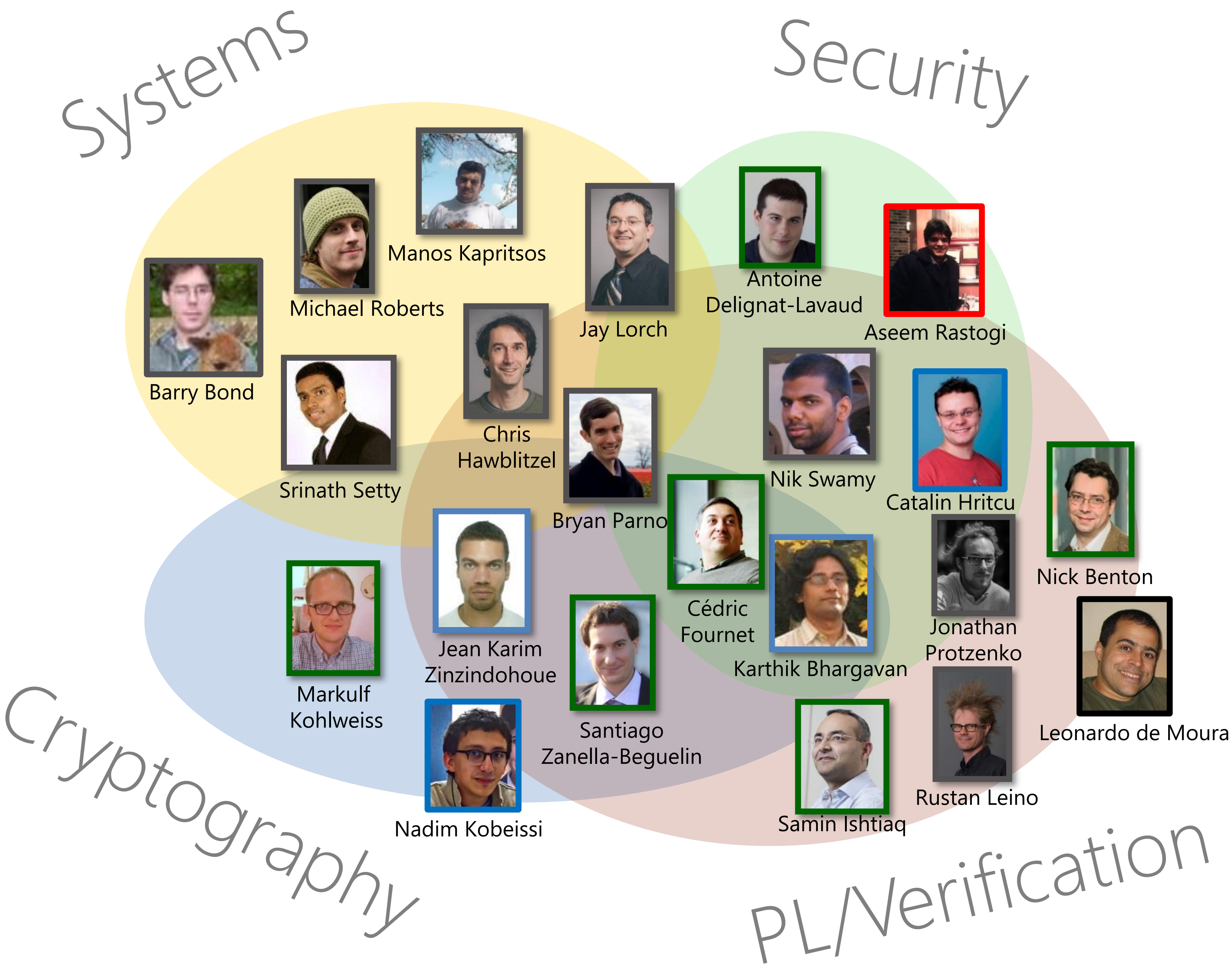


Everest Goals

- Strong verified security
- Widespread deployment
- Trustworthy usable tools



Everest Team



Microsoft Research

- Cambridge
- Redmond
- Bangalore

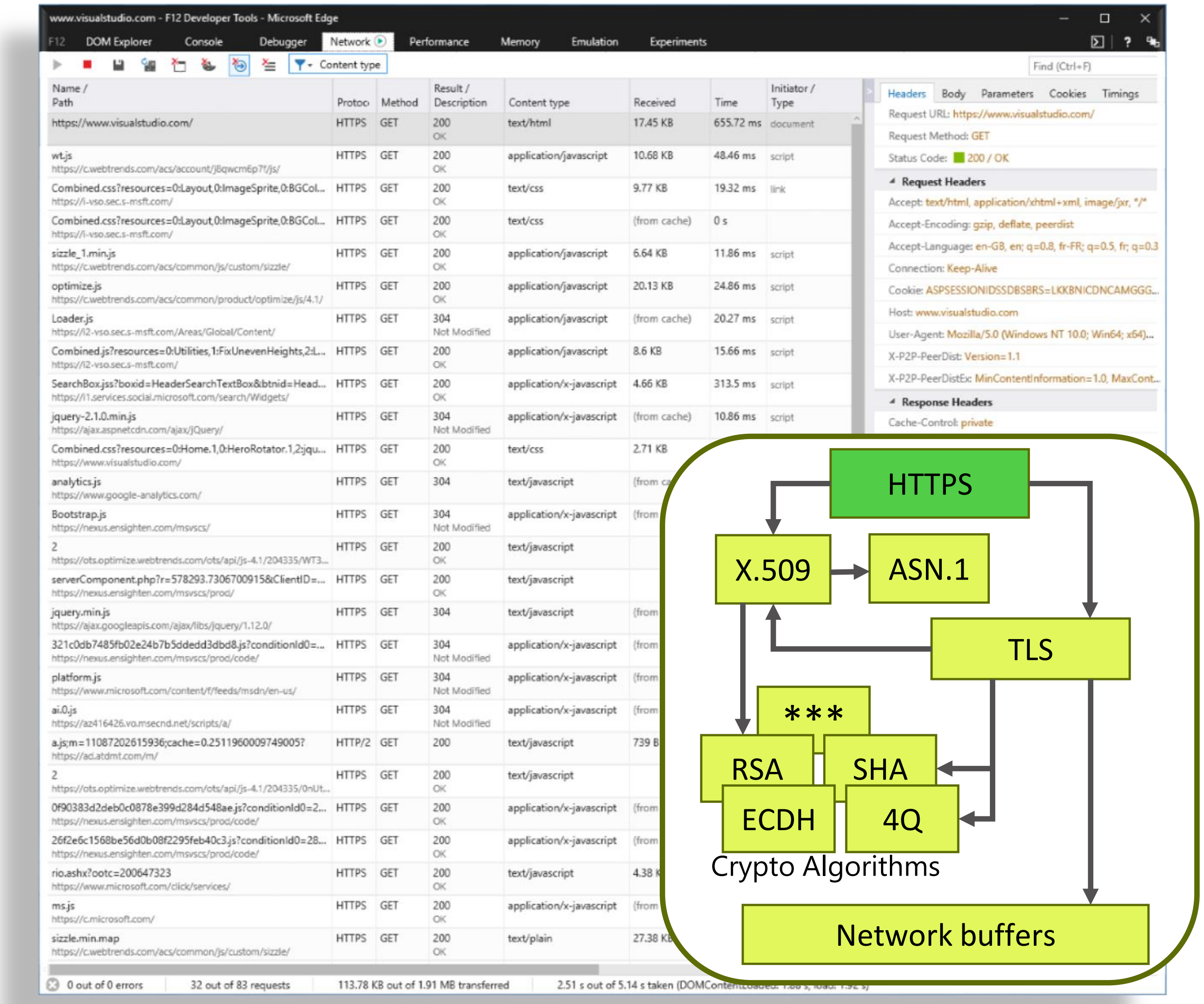
Paris

Application Security: https://

Demo: tracing

https://www.visualstudio.com/

- **Trust is transitive**
  each page involves connections
  to many servers (different origins)
- **Trust is implicit**
  17 concurrent TLS connections,
  configurations, certificate chains
- **Trust is a matter of state**
  cookies, caches, configurations, proxies



Long-term identities: X.509

Public-Key Infrastructure (Certificate Chains)

Designed in 1984; widely criticized but hard to replace
HTTPS is just one application

Same complexity as TLS?

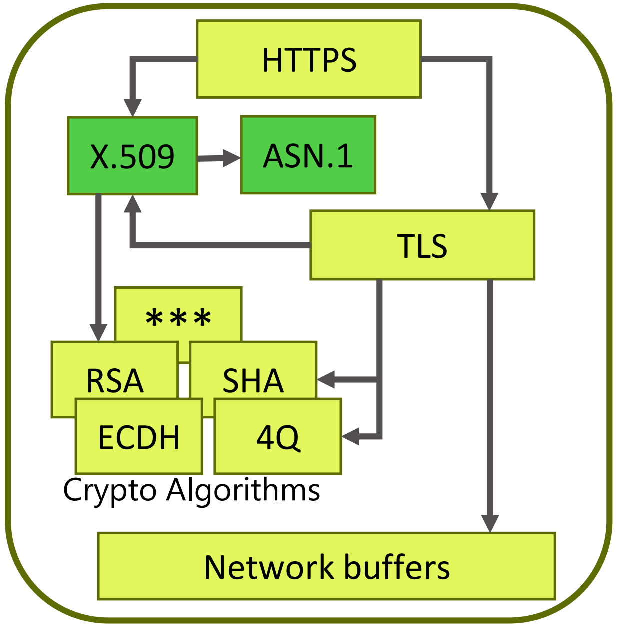
ASN.1 grammar; many extensions and interpretations
50% of "TLS attacks" are in fact X.509 attacks

Recent initiatives

Global scans for millions of certificates
Certificate pinning & transparency
Let's encrypt! https://letsencrypt.org/

Verification?

Complex ambiguous format
Certificate issuance and revocation policies



Cryptographic Algorithms for HTTPS

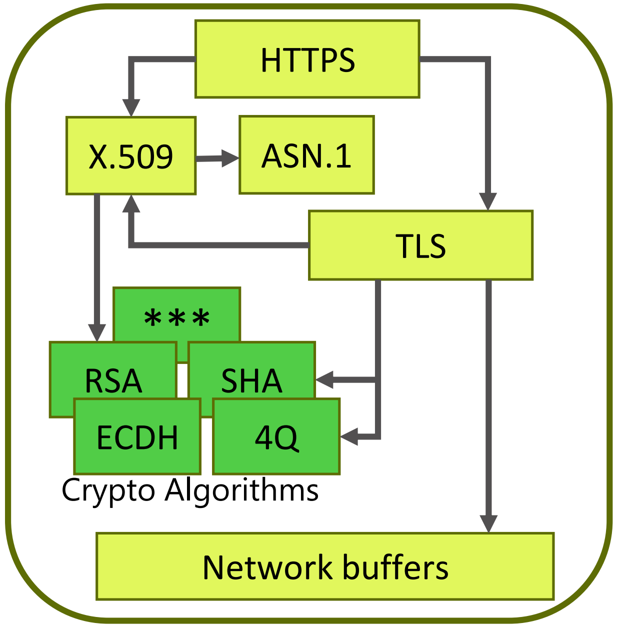
Algorithms get broken & replaced over time

Security relies on probabilistic cryptographic assumptions (who knows?)

Modern design & implementations select between
various algorithms & implementations for the same core functionality

~30 standard algorithms

- Hash and key-derivation functions (SHA256)
- Symmetric cryptography (AES_GCM, AES_CBC)
- Public-key encryption and signing
- Elliptic curves (NIST, 25519, 4Q)

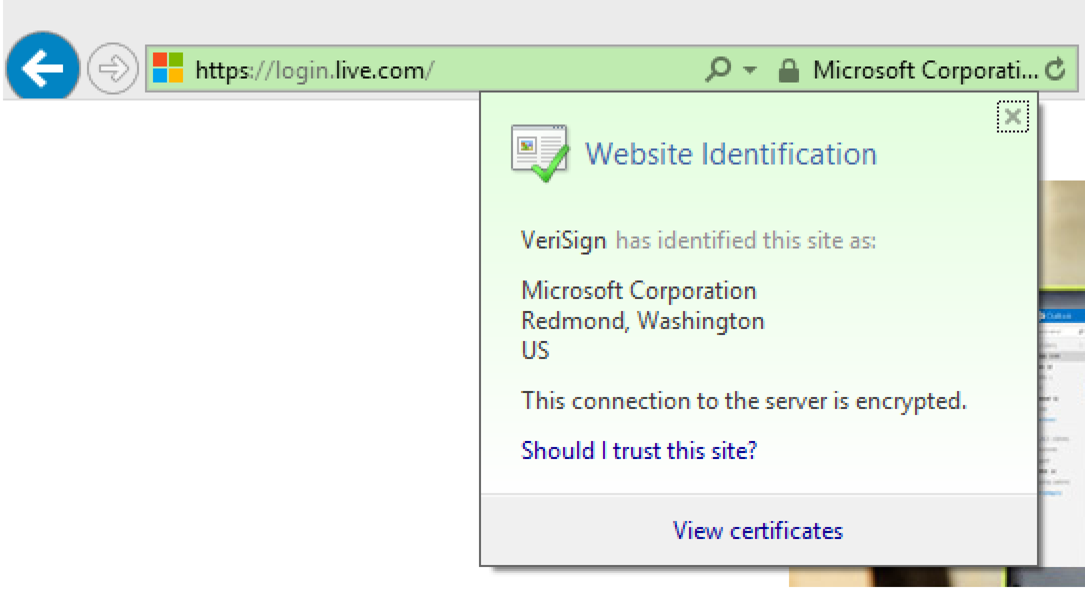


High-performance

AES_GCM takes 0.46 cycle/byte on Intel Skylake
Hand-tuned, low-level, architecture-specific

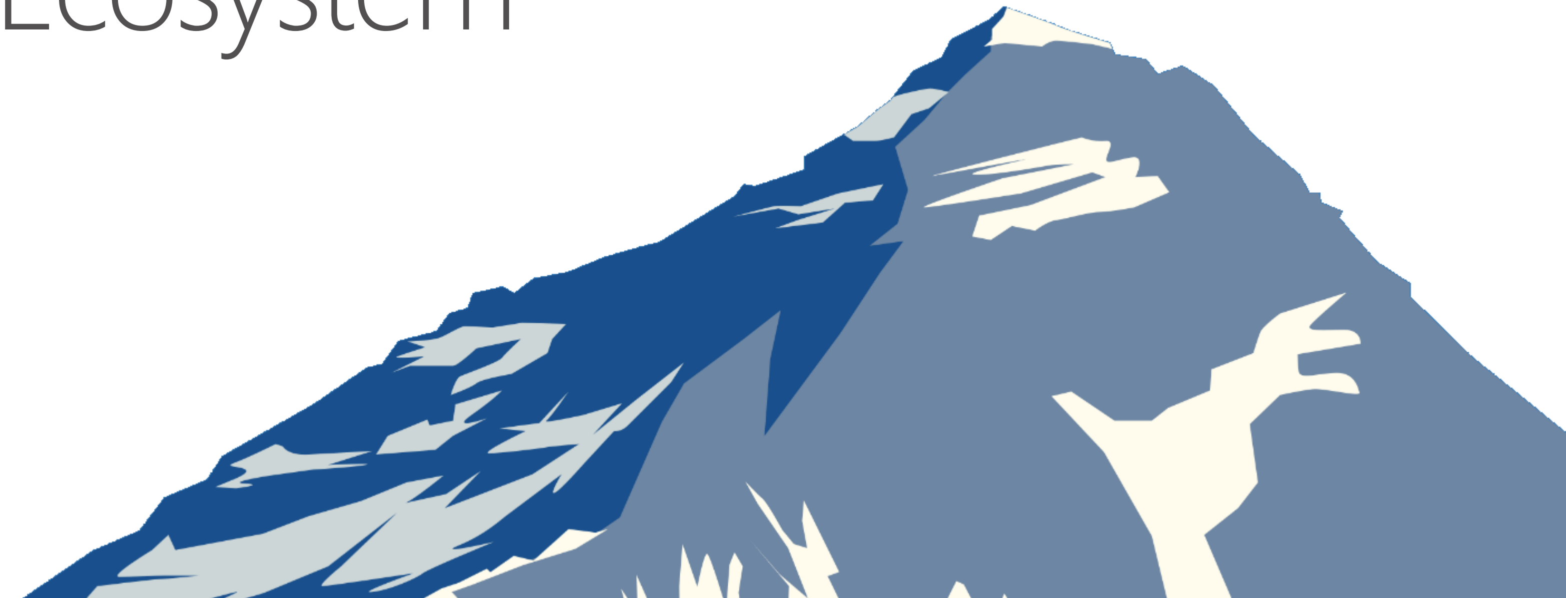
Microsoft Research

Microsoft Research - Inria
JOINT CENTRE



Verified Secure Implementations for the HTTPS Ecosystem

miTLS & Everest*



*the Everest VERified End-to-end Secure Transport