University of Glasgow

Internet Protocols Laboratory

Secure Communications

Networked Systems (H)

Lecture 3

Colin Perkins | https://csperkins.org/ | Copyright © 2026 University of Glasgow | This work is licensed under the Creative Commons Attribution-NoDerivatives 4.0 International License.
To view a copy of this license, visit http://creativecommons.org/licenses/by-nd/4.0/ or send a letter to Creative Commons, PO Box 1866, Mountain View, CA 94042, USA.

Lecture Outline

- The Need for Secure Communications
- Principles of Secure Communication
- Transport Layer Security (TLS) v1.3
- Discussion

Colin Perkins | https://csperkins.org/ | Copyright © 2026 University of Glasgow

University of Glasgow

The Need for Secure Communications

- Network Traffic Monitoring
- Protecting Privacy
- Protecting Message Integrity
- Preventing Protocol Ossification

CC BY ND

Many Organisations Monitor Internet traffic

- Governments, intelligence agencies, and law enforcement
  - Good reasons to monitor **some** traffic, but Edward Snowden showed pervasive monitoring of **all** traffic – no reason to believe this has stopped
- Businesses
  - Regulatory requirements to record some traffic – e.g., banking
  - To profile customers for advertising
- Network operators
  - To support network operations and trouble-shooting
  - To profile customers for advertising
- Criminals and malicious users
  - Identity theft, vulnerability scanning



Edward Snowden

S. Farrell and H. Tschofenig, “Pervasive Monitoring is an Attack”, IETF, May 2014, RFC 7258 https://datatracker.ietf.org/doc/rfc7258/

K. Moriarty and A. Morton, “Effect of Pervasive Encryption on Operators”, IETF, July 2018, RFC 8404 https://datatracker.ietf.org/doc/rfc8404/

G. Fairhurst and C. Perkins, “Considerations around Transport Header Confidentiality, Network Operations, and the Evolution of Internet Transport Protocols”, IETF,July 2021, RFC 9065, https://datatracker.ietf.org/doc/rfc9065/

Colin Perkins | https://csperkins.org/ | Copyright © 2026 University of Glasgow

Protecting Privacy

- Some network monitoring/surveillance is beneficial and desirable, some monitoring is malicious and actively harmful
- **Mechanisms that protect privacy against malicious attackers will also prevent benign monitoring by law enforcement**
- Mechanisms that permit access by law enforcement risk being repurposed to permit access by attackers
  - Example: the Salt Typhoon hack of US telecommunications providers, allegedly by China, was a breach of the systems used by US law enforcement to conduct court-authorised surveillance

Colin Perkins | https://csperkins.org/ | Copyright © 2026 University of Glasgow

Protecting Message Integrity

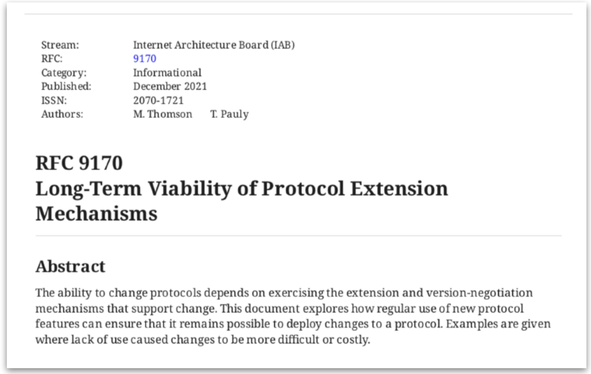
- Numerous organisations may want to change messages in transit
  - Governments, intelligence agencies, and law enforcement
    - Many governments require ISPs to censor or modify DNS responses
    - Many governments require ISPs to censor messages containing certain content
  - Businesses and network operators
    - To enforce legal restrictions on content, terms of service, or to prevent copyright infringement
  - Criminals and malicious users
    - Phishing scams and identity theft
- Mechanisms to protect messages from malicious attackers also prevent benign actors
  - e.g., web browsers deploy DNS-over-HTTPS (DoH) to ensure integrity of DNS responses → protects against phishing attacks that modify DNS replies → stops ISPs modifying DNS responses to prevent access to domains hosting illegal material
- **No technical way to distinguish beneficial integrity violations from malicious attacks**

Colin Perkins | https://csperkins.org/ | Copyright © 2026 University of Glasgow

Preventing Protocol Ossification

- Network operators deploy **middleboxes** to monitor or control traffic
  - NATs, firewalls, monitors, traffic shapers, filters, …
  - These middleboxes must understand the protocols used – they can't monitor or control traffic that they don't understand

- Increasing use of middleboxes causes **protocol ossification**
  - cannot change end-to-end protocols, because doing so interacts poorly with middleboxes that don't understand the changes
  - The more of a protocol that is encrypted, the easier it is to change – since middleboxes can't understand or modify the data – but the harder it is for middleboxes to provide useful services



Stream: Internet Architecture Board (IAB)
RFC: 9170
Category: Informational
Published: December 2021
ISSN: 2070-1721
Authors: M. Thomson T. Pauly

RFC 9170
Long-Term Viability of Protocol Extension Mechanisms

Abstract

The ability to change protocols depends on exercising the extension and version-negotiation mechanisms that support change. This document explores how regular use of new protocol features can ensure that it remains possible to deploy changes to a protocol. Examples are given where lack of use caused changes to be more difficult or costly.

M. Thomson and T. Pauly, "Long-term Viability of Protocol Extension Mechanisms", Internet Architecture Board, December 2021, RFC 9170. https://datatracker.ietf.org/doc/rfc9170/

Colin Perkins | https://csperkins.org/ | Copyright © 2026 University of Glasgow

Security and Policy: Encryption and Key Escrow

- Strong reasons to protect privacy, prevent modification of data in transit, and prevent protocol ossification using end-to-end encryption and authentication
- But doing so can complicate the job of law enforcement and those performing legitimate network monitoring – numerous proposals for key escrow, targeted weakening of end-to-end encryption, and other forms of exceptional access
- Dialogue between engineers, protocol designers, operators, policy makers, and law enforcement crucial – to understand constraints and concerns



Keys Under Doormats:

MANDATING INSECURITY BY REQUIRING GOVERNMENT ACCESS TO ALL DATA AND COMMUNICATIONS

Harold Abelson, Ross Anderson, Steven M. Bellovin, Josh Benaloh, Matthew Blaze, Whitfield Diffie, John Gilmore, Matthew Green, Peter G. Neumann, Susan Landau, Ronald L. Rivest, Jeffrey I. Schiller, Bruce Schneier, Michael Specter, Daniel J. Weitzner

Abstract

Twenty years ago, law enforcement organizations lobbied to require data and communication services to engineer their products to guarantee law enforcement access to all data. After lengthy debate and vigorous predictions of enforcement channels "going dark," these attempts to regulate the emerging Internet were abandoned. In the intervening years, innovation on the Internet flourished, and law enforcement agencies found new and more effective means of accessing vastly larger quantities of data. Today we are again hearing calls for regulation to mandate the provision of exceptional access mechanisms. In this report, a group of computer scientists and security experts, many of whom participated in a 1997 study of these same topics, has convened to explore the likely effects of imposing extraordinary access mandates.

We have found that the damage that could be caused by law enforcement exceptional access requirements would be even greater today than it would have been 20 years ago. In the wake of the growing economic and social cost of the fundamental insecurity of today's Internet environment, any proposals that alter the security dynamics online should be approached with caution. Exceptional access would force Internet system developers to reverse "forward secrecy" design practices that seek to minimize the impact on user privacy when systems are breached. The complexity of today's Internet environment, with millions of apps and globally connected services, means that new law enforcement requirements are likely to introduce unanticipated, hard to detect security flaws. Beyond these and other technical vulnerabilities, the prospect of globally deployed exceptional access systems raises difficult problems about how such an environment would be governed and how to ensure that such systems would respect human rights and the rule of law.

H. Abelson, *et al.*, "Keys under doormats: Mandating insecurity by requiring government access to all data and communications", MIT Computer Science and Artificial Intelligence Lab, technical report MIT-CSAIL-TR-2015-026, July 2015. http://dspace.mit.edu/handle/1721.1/97690

Colin Perkins | https://csperkins.org/ | Copyright © 2026 University of Glasgow

Security and Policy: End-system Content Monitoring

- If data is encrypted end-to-end, protecting data in transit from malicious attacks, should law enforcement be able to monitor traffic at the end-points?
- Does the answer change if both end-points are personal devices vs. if one of the end-points is a data centre?
- How do the expectations for privacy, law enforcement access, and abuse protection differ between one-to-one conversations, group conversations, social networks, and other situations?



arXiv:2110.07450v1 [cs.CR] 14 Oct 2021

Bugs in our Pockets:
The Risks of Client-Side Scanning

Hal Abelson Ross Anderson Steven M. Bellovin
Josh Benaloh Matt Blaze Jon Callas Whitfield Diffie
Susan Landau Peter G. Neumann Ronald L. Rivest
Jeffrey I. Schiller Bruce Schneier Vanessa Teague
Carmela Troncoso

October 15, 2021

Executive Summary

Our increasing reliance on digital technology for personal, economic, and government affairs has made it essential to secure the communications and devices of private citizens, businesses, and governments. This has led to pervasive use of cryptography across society. Despite its evident advantages, law enforcement and national security agencies have argued that the spread of cryptography has hindered access to evidence and intelligence. Some in industry and government now advocate a new technology to access targeted data: *client-side scanning* (CSS). Instead of weakening encryption or providing law enforcement with backdoor keys to decrypt communications, CSS would enable on-device analysis of data in the clear. If targeted information were detected, its existence and, potentially, its source, would be revealed to the agencies; otherwise, little or no information would leave the client device. Its proponents claim that CSS is a solution to the encryption versus public safety debate: it offers privacy—in the sense of unimpeded end-to-end encryption—and the ability to successfully investigate serious crime.

In this report, we argue that CSS neither guarantees efficacious crime prevention nor prevents surveillance. Indeed, the effect is the opposite. CSS by its nature creates serious security and privacy risks for all society while the assistance it can provide for law enforcement is at best problematic. There are multiple ways in which client-side scanning can fail, can be evaded, and can be abused.

Its proponents want CSS to be installed on all devices, rather than installed covertly on the devices of suspects, or by court order on those of ex-offenders. But universal deployment threatens the security of law-abiding citizens as well as law-breakers. Technically, CSS allows end-to-end encryption, but this is moot if the

1

H. Abelson, R. Anderson, S. M. Bellovin, J. Benaloh, M. Blaze, J. Callas, W. Diffie, S. Landau, P. G. Neumann, R. Rivest, J. I. Schiller, B. Schneier, V. Teague, and C. Troncosco. "Bugs in our pockets: The risks of client-side scanning". October 2021. https://arxiv.org/abs/2110.07450

Colin Perkins | https://csperkins.org/ | Copyright © 2026 University of Glasgow

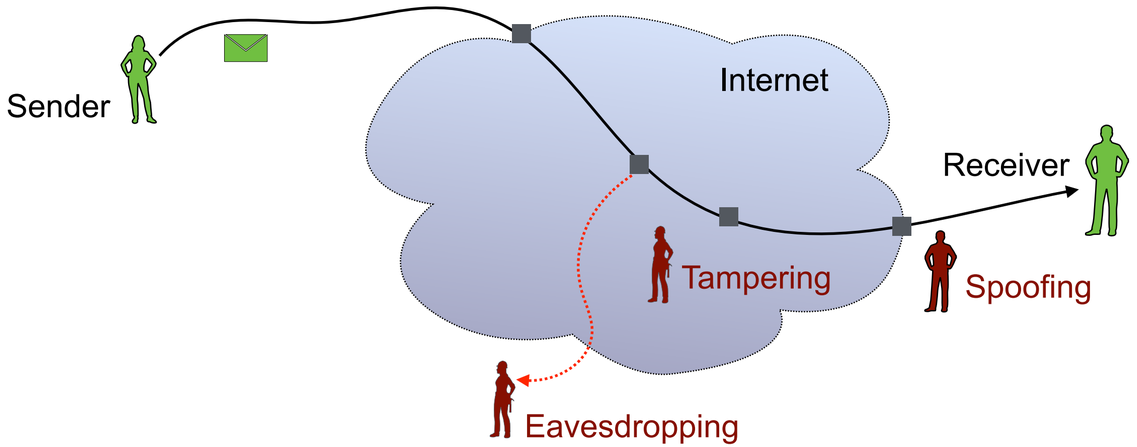
University of Glasgow

Principles of Secure Communication

- Ensuring Confidentiality
- Authenticating Messages
- Validating Identity

Colin Perkins | https://csperkins.org/ | Copyright © 2026 University of Glasgow

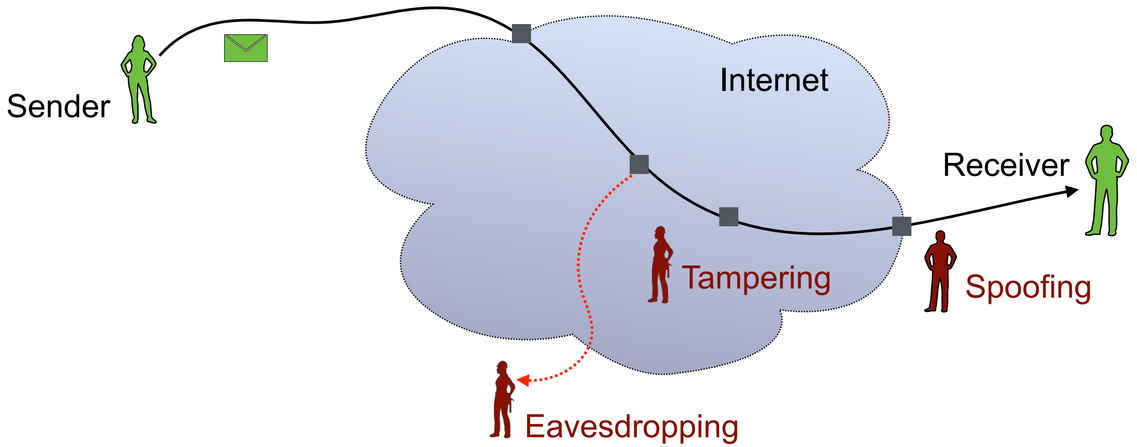
Goals of Secure Communication



- Goal: deliver message from sender to receiver
  - Avoid eavesdropping → encrypt to provide confidentiality
  - Avoid tampering → authenticate to ensure the message is not modified in transit
  - Avoid spoofing → validate identity of sender

Colin Perkins | https://csperkins.org/ | Copyright © 2026 University of Glasgow

How to Provide Confidentiality? (1/2)



- Data traversing the network can be read by any device on the path
  - Can eavesdrop on packets as they traverse a link
  - Configure a switch or router to snoop on data as it's forwarded between links
- The network operator can **always** do this; if their network has been compromised, maybe so can others

Colin Perkins | https://csperkins.org/ | Copyright © 2026 University of Glasgow

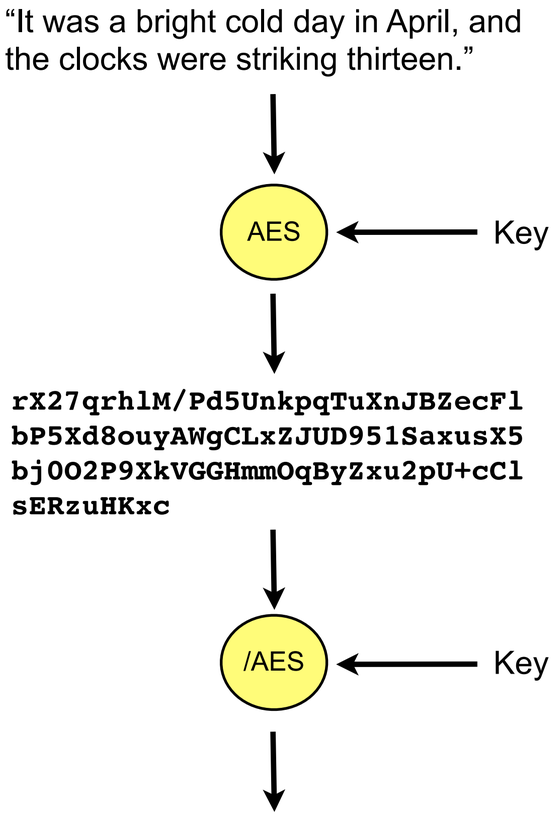
How to Provide Confidentiality? (2/2)

- Data can **always** be read → **use encryption** to make it useless if intercepted
- Two basic approaches
  - Symmetric cryptography
    - Advanced Encryption Standard (AES)
  - Public key cryptography
    - The Diffie-Hellman algorithm
    - The Rivest-Shamir-Adleman (RSA) algorithm
    - **Elliptic curve-based algorithms**
  - Complex mathematics – will not attempt to describe

Colin Perkins | https://csperkins.org/ | Copyright © 2026 University of Glasgow

Symmetric Cryptography

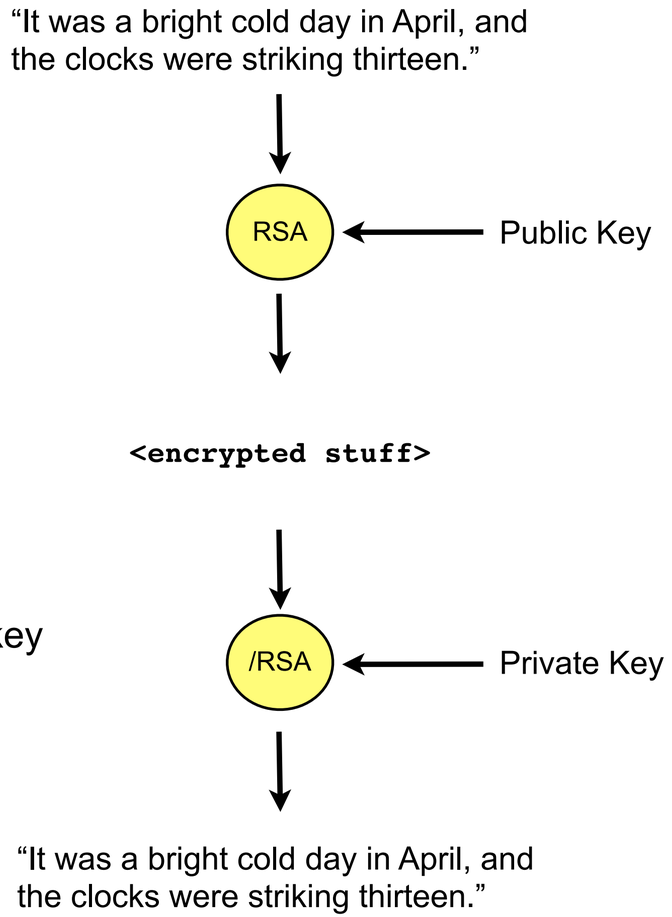
- Symmetric encryption converts plain text into cipher-text
  - A secret **key** control the encryption and decryption process
    - Same key used to encrypt as is used to decrypt
    - Provided the key is secret and only known to sender and receiver, the conversation is secure – problem: **how to securely distribute the key?**
  - Very fast – suitable for bulk encryption
  - Encryption and decryption algorithms are public
    - US Advanced Encryption Standard (AES) is widely used – based on Rijndael algorithm, developed by Vincent Rijmen and Joan Daemen https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.197.pdf



Colin Perkins | https://csperkins.org/ | Copyright © 2026 University of Glasgow

Public Key Cryptography

- Public key encryption also converts plain text into cipher-text
  - The algorithms are public: Diffie–Hellman, Rivest–Shamir–Adleman, Ephemeral Elliptic Curve Diffie-Hellman, …
  - Public key algorithms use two related keys:
    - The **public key** for a user is widely distributed
    - **The corresponding private key must be kept secret**
    - If one key is used to encrypt, the other key is needed to decrypt the message
  - Solves key distribution problem
    - Look-up the public key of the receiver in a directory
    - Sender uses the public key to encrypt the message → can only be decrypted by the private key
    - If receiver is trusted to keep private key secret, only it can decrypt the message
- Problem: **very slow** to encrypt and decrypt



Colin Perkins | https://csperkins.org/ | Copyright © 2026 University of Glasgow

Hybrid Cryptography

- Modern communications use a combination of **both** public-key and symmetric cryptography for security and speed
  - Sender chooses a random value, K_s, that can be used as key for the symmetric encryption algorithm
  - Sender looks up the receiver's public key, K_{pub}, uses it to encrypt K_s, and sends the result to the receiver; receiver uses the corresponding private key, K_{priv}, to decrypt the message and retrieve K_s
    - Securely transfers K_s from sender to receiver
    - Public key encryption is very slow, but the key K_s is small, so this doesn't matter
  - Sender encrypts future messages using symmetric cryptography with key K_s, receiver also has k_s, which it uses to decrypt the messages
    - Symmetric cryptography is fast, but requires the key to be exchanged securely
    - The public key algorithm has been used to securely exchange the key
- Ensures confidentiality of communication with good performance

Colin Perkins | https://csperkins.org/ | Copyright © 2026 University of Glasgow

Authentication

- Encryption can ensure confidentiality – but also need to verify identify of sender and ensure messages have not been modified in transit
  - Generate a **digital signature** to authenticate the message
  - Relies on public key cryptographic and a cryptographic hash

Colin Perkins | https://csperkins.org/ | Copyright © 2026 University of Glasgow

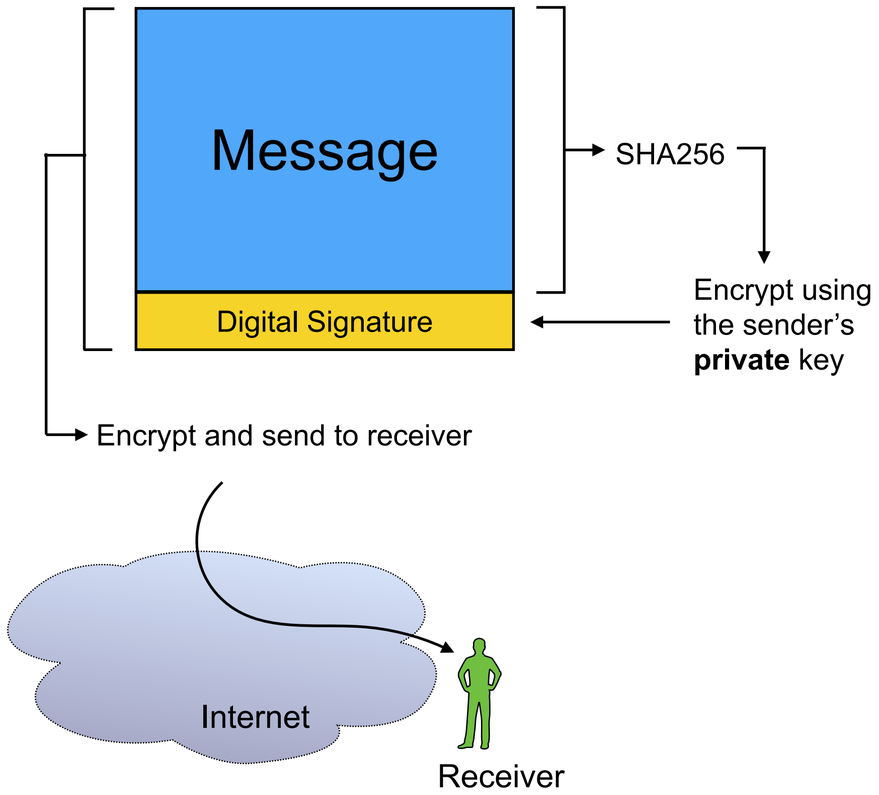
Cryptographic Hash Functions



- **Cryptographic hash** takes arbitrary input and produces a fixed length output hash
  - Any change to input generates different output
  - Infeasible to find two inputs that give the same output
  - Calculating a cryptographic hash is fast
  - Reversing a hash, to find the input given only the output, is infeasible
- Many cryptographic hash algorithms exist:
  - Recommendation: SHA256 algorithm https://datatracker.ietf.org/doc/rfc6234/
  - Older algorithms, e.g., MD5 and SHA1, have known security flaws

Colin Perkins | https://csperkins.org/ | Copyright © 2026 University of Glasgow

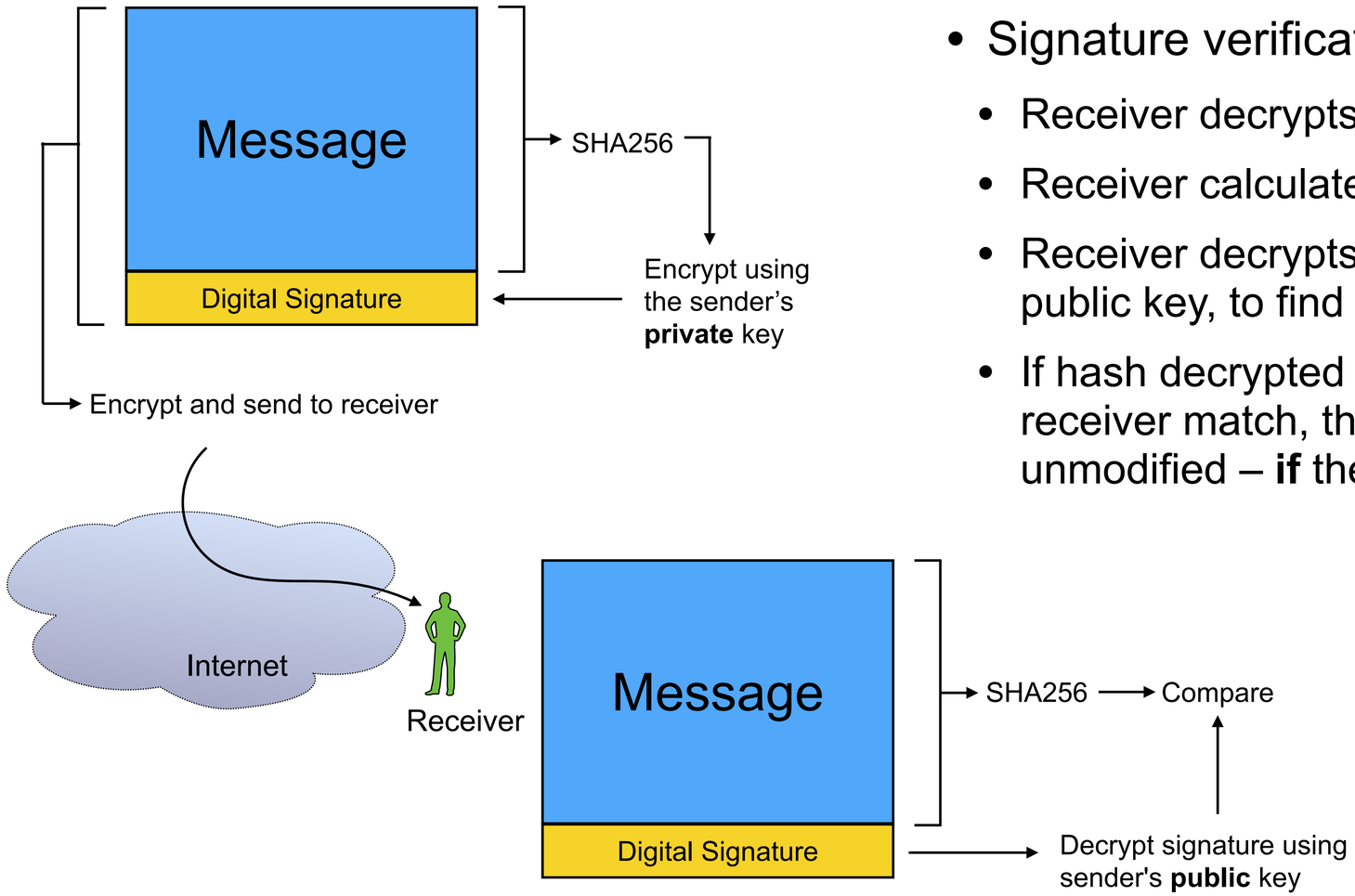
Digital Signatures (1/2)



- Sender generates a digital signature
  - Sender calculates cryptographic hash of the message
  - Sender encrypts the hash with their own **private** key
    - Anyone can use the sender's public key to decrypt this, but only the sender can have encrypted it (if trusted to keep their private key secret)
  - Attaches encrypted hash to the message
- Message and its digital signature are encrypted and sent to receiver using hybrid encryption

Colin Perkins | https://csperkins.org/ | Copyright © 2026 University of Glasgow

Digital Signatures (2/2)

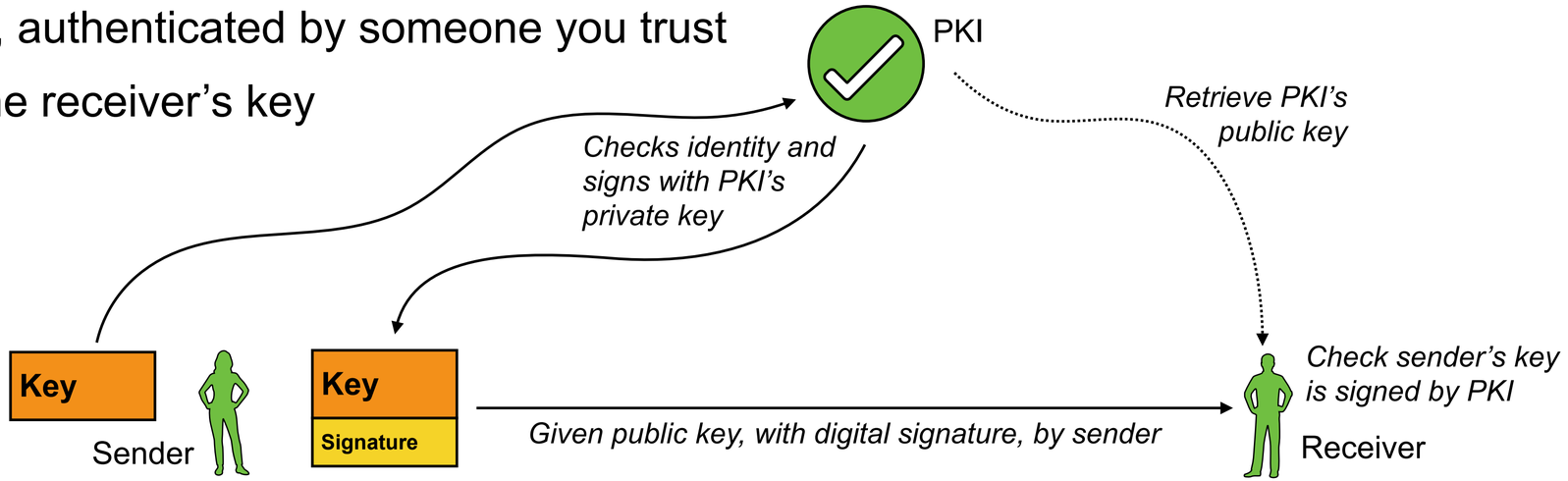


- Signature verification process:
  - Receiver decrypts the message
  - Receiver calculate cryptographic hash of the message
  - Receiver decrypts digital signature using sender's public key, to find the hash that the sender calculates
  - If hash decrypted hash and the hash calculated by the receiver match, then the message is authentic and unmodified – **if** the sender kept its private key secret

Colin Perkins | https://csperkins.org/ | Copyright © 2026 University of Glasgow

Trust and Public Key Infrastructure

- How to know what public key corresponds to a particular receiver?
  - The receiver gave you their key in person
  - The receiver sent you their key, authenticated by someone you trust
  - Someone you trust gave you the receiver's key



- A **public key infrastructure** can authenticate keys
  - PKI verifies sender's identity, then adds their digital signature to sender's public key
  - If receiver trusts PKI, can verify the digital signature to confirm identity of sender

Colin Perkins | https://csperkins.org/ | Copyright © 2026 University of Glasgow

University of Glasgow

Transport Layer Security (TLS) v1.3

- What is TLS?
- TLS handshake protocol
- TLS record protocol
- Limitations

CC BY ND

Transport Layer Security (TLS) v1.3

- TCP is not secure – neither TCP headers nor the data are encrypted or authenticated
- **Transport Layer Security** (TLS v1.3) can be used to encrypt and authenticate data carried **within** a TCP connection
  - Official specification:
    - https://datatracker.ietf.org/doc/html/rfc8446 – The Transport Layer Security (TLS) Protocol Version 1.3
  - Useful blog posts:
    - https://ietf.org/blog/tls13/ – Introduction to TLS 1.3 from IETF
    - https://blog.cloudflare.com/rfc-8446-aka-tls-1-3/ – A detailed look at what's new in TLS 1.3
    - https://davidwong.fr/tls13/ – The TLS 1.3 specification, redrawn in a readable way with explanatory videos and comments
    - https://tls13.ulfheim.net – An annotated packet capture, showing details of a TLS Connection

Colin Perkins | https://csperkins.org/ | Copyright © 2026 University of Glasgow

TLS v1.3 Goals

> The primary goal of TLS is to provide a secure channel between two communicating peers; the only requirement from the underlying transport is a reliable, in-order data stream. Specifically, the secure channel should provide the following properties:
>
> - **Authentication:** The server side of the channel is always authenticated; the client side is optionally authenticated. Authentication can happen via asymmetric cryptography (e.g., RSA [RSA], the Elliptic Curve Digital Signature Algorithm (ECDSA) [ECDSA], or the Edwards-Curve Digital Signature Algorithm (EdDSA) [RFC8032]) or a symmetric pre-shared key (PSK).
>
> - **Confidentiality:** Data sent over the channel after establishment is only visible to the endpoints. TLS does not hide the length of the data it transmits, though endpoints are able to pad TLS records in order to obscure lengths and improve protection against traffic analysis techniques.
>
> - **Integrity:** Data sent over the channel after establishment cannot be modified by attackers without detection.

E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.3", IETF, August 2018, RFC 8446, https://datatracker.ietf.org/doc/html/rfc8446

Colin Perkins | https://csperkins.org/ | Copyright © 2026 University of Glasgow

TLS v1.3 Overview

> TLS consists of two primary components:
>
> - A handshake protocol (Section 4) that authenticates the communicating parties, negotiates cryptographic modes and parameters, and establishes shared keying material. The handshake protocol is designed to resist tampering; an active attacker should not be able to force the peers to negotiate different parameters than they would if the connection were not under attack.
>
> - A record protocol (Section 5) that uses the parameters established by the handshake protocol to protect traffic between the communicating peers. The record protocol divides traffic up into a series of records, each of which is independently protected using the traffic keys.

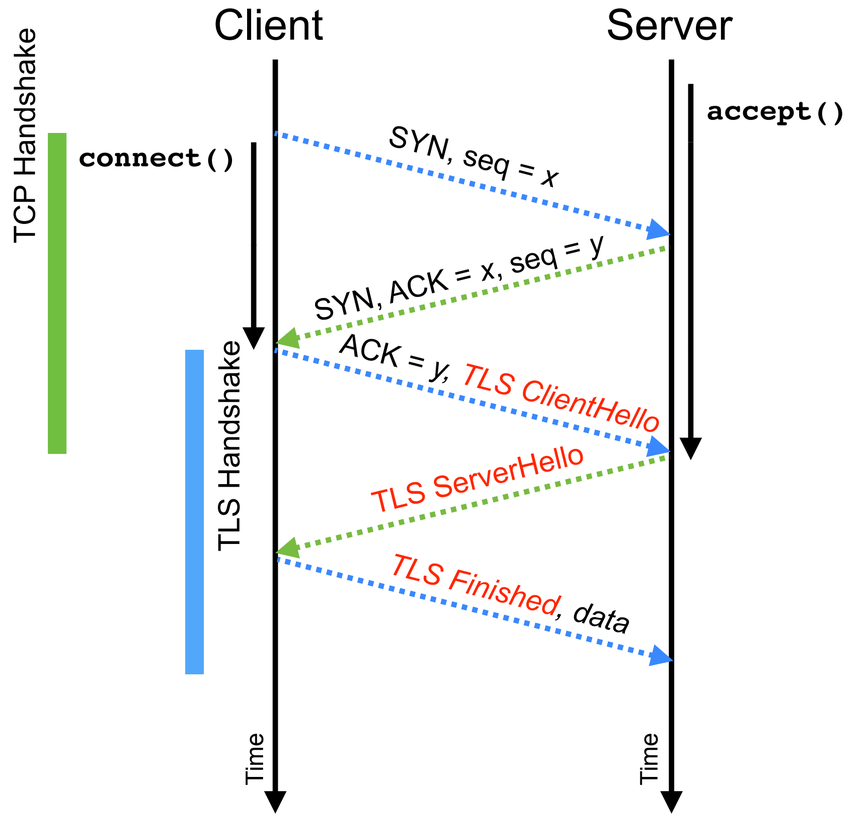
E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.3", IETF, August 2018, RFC 8446, https://datatracker.ietf.org/doc/html/rfc8446

- A TCP connection is established
- TLS runs within that connection
  - TLS handshake protocol authenticates endpoints and agrees encryption keys
  - TLS record protocol allows endpoints to deliver authenticated and encrypted blocks of data
  - TLS record protocol also provides framing – splits the TCP byte stream into a series of records

Colin Perkins | https://csperkins.org/ | Copyright © 2026 University of Glasgow

TLS v1.3 Handshake Protocol

- TCP connection established as usual
  - **`SYN`** → **`SYN+ACK`** → **`ACK`**
- TLS handshake protocol immediately follows
  - TLS **`ClientHello`** sent with the **`ACK`**
  - TLS **`ServerHello`** sent in response
  - TLS **`Finished`** message concludes, and carries initial secure data record
- Adds 1-RTT to connection establishment



Colin Perkins | https://csperkins.org/ | Copyright © 2026 University of Glasgow

TLS v1.3 `ClientHello`

> - **Key Exchange: Establish shared keying material and select the cryptographic parameters. Everything after this phase is encrypted.**

E. Rescorla, “The Transport Layer Security (TLS) Protocol Version 1.3”, IETF, August 2018, RFC 8446, https://datatracker.ietf.org/doc/html/rfc8446

- The **`ClientHello`** message:
  - Indicates that TLS v1.3 is to be used
    - Signals TLS **v1.2** in the main **`ClientHello`** message, with an extension header to say “I’m really TLS v1.3” because too many middleboxes break if the TLS version changes – *protocol ossification*
  - Provides the cryptographic algorithms the client supports
  - Provides the name of the server to which the client is connecting
    - If connecting to a web hosting server, it’s likely that more than one site is hosted on the same server, so need to specify which is intended
  - Does not contain any data

Colin Perkins | https://csperkins.org/ | Copyright © 2026 University of Glasgow

TLS v1.3 `ServerHello`

> - Server Parameters: Establish other handshake parameters (whether the client is authenticated, application-layer protocol support, etc.).
>
> The server processes the ClientHello and determines the appropriate cryptographic parameters for the connection. It then responds with its own ServerHello (Section 4.1.3), which indicates the negotiated connection parameters. The combination of the ClientHello and the ServerHello determines the shared keys

E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.3", IETF, August 2018, RFC 8446, https://tools.ietf.org/html/rfc8446

- The **`ServerHello`** message:
  - Indicates that TLS v1.3 is to be used
    - It signals TLS **v1.2** in the main **`ServerHello`** message, and includes an extension header to say "I'm really TLS v1.3" because too many middleboxes break if the TLS version changes – **ossification**
  - Provides cryptographic algorithms selected by the server, from the set the client suggested
  - Provides the server's public key and digital signature used to verify its identity
  - Does not contain any data

Colin Perkins | https://csperkins.org/ | Copyright © 2026 University of Glasgow

TLS v1.3 `Finished`

> - **Authentication: Authenticate the server (and, optionally, the client) and provide key confirmation and handshake integrity.**

E. Rescorla, “The Transport Layer Security (TLS) Protocol Version 1.3”, IETF, August 2018, RFC 8446, https://tools.ietf.org/html/rfc8446

- The **`Finished`** message:
  - Provides the client’s public key
  - Optionally, provides the certificate needed to authenticate the client to the server
  - May contain data sent from client to server

Colin Perkins | https://csperkins.org/ | Copyright © 2026 University of Glasgow

TLS v1.3 Cryptographic Algorithms

- TLS uses Ephemeral Elliptic Curve Diffie-Hellman (ECDHE) key exchange
  - Client sends its public key to server in **`ClientHello`**
  - Server sends its public key to client in **`ServerHello`**
    - The public keys are *ephemeral* – the client and server both generate new, randomly chosen, private keys for each connection and derive the public keys from them
  - Client and server both combine the two public keys with their own private key to derive the key used for the symmetric cryptography – complex mathematics; will not describe
  - Server provides a certificate (digital signature) to allow the client to verify its identify in the **`ServerHello`** – client can optionally provide this in **`Finished`**
- Client proposes symmetric encryption algorithms in **`ClientHello`**, server picks from them and replies in **`ServerHello`**
  - Either AES or ChaCha20 symmetric encryption

Colin Perkins | https://csperkins.org/ | Copyright © 2026 University of Glasgow

TLS v1.3 Record Protocol

- TLS v1.3 splits the data into **records**, each containing ≤ 2^{14} bytes
  - Each record is both encrypted and authenticated; it has a sequence number
  - Can renegotiate encryption keys between records
  - TCP does not preserve record boundaries; TLS adds **framing** so that it does – reading from a TLS connection will block until a complete record is received

- Client and server exchange records – send and receive data – then close connection

Colin Perkins | https://csperkins.org/ | Copyright © 2026 University of Glasgow

Limitations of TLS

- **TLS is secure**, **but** has limitations:
  - Operates within a TCP/IP connection:
    - IP addresses and TCP port numbers are not protected – exposes information about who is communicating and what application is being used
  - The TLS handshake **does not encrypt the server name:**
    - Exposes name of server to which connection is being made in plain text – **is this a real concern?**
    - An encrypted SNI extension is in development
  - Relies on a PKI to validate public keys:
    - **Potential concerns about trustworthiness of the PKI**
    - Who runs the PKI and certificate authorities and are they trustworthy?

Colin Perkins | https://csperkins.org/ | Copyright © 2026 University of Glasgow

University of Glasgow

Discussion

- End-to-end security
- Robustness principle
- Validating input data
- Writing secure code

CC BY ND

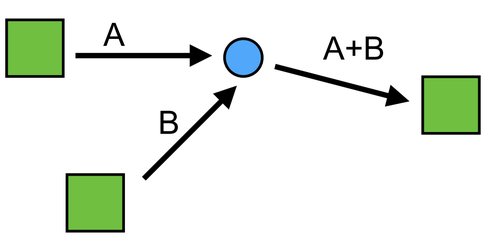
End-to-end Security? (1/2)

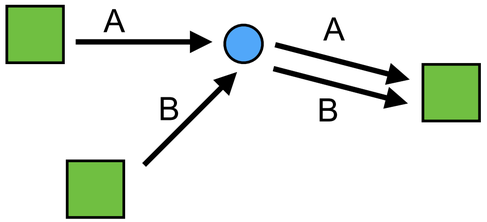
- For communication to be secure, it must be **end-to-end**
- The two endpoints are the sender and the final recipient
  - With a data centre or CDN, what is the final recipient?
    - Is it the load balancer at the entrance to the data centre, or the server within the data centre that processes the request?
    - If the request is directed to a content distribution network (CDN), is the end the local cache that serves the request? If so, how are the encryption keys shared?
  - If the data is moving between two users, is it encrypted between the two users or between each user and the data centre?
    - i.e., can the data centre see user-to-user data flows?
    - (This is normal if you run TLS from user-to-data centre then from data centre-to-user)

Colin Perkins | https://csperkins.org/ | Copyright © 2026 University of Glasgow

End-to-end Security? (2/2)

- Is there in-network processing? How much data is revealed to the in-network server?
  - e.g., video conference with privacy protection vs. without
  - Does the central server decrypt the speech data, mix into one stream and send to the receiver, or does it forward all active streams in encrypted form





  - Trades-off security vs bandwidth
    - For audio the bandwidth is small enough this doesn't matter
    - For video conferencing, the combined bandwidth may be significant

Colin Perkins | https://csperkins.org/ | Copyright © 2026 University of Glasgow

The Robustness Principle (Postel's Law)

```
At every layer of the protocols, there is a general rule whose
application can lead to enormous benefits in robustness and
interoperability:

       "Be liberal in what you accept, and
        conservative in what you send"

Software should be written to deal with every conceivable
error, no matter how unlikely; sooner or later a packet will
come in with that particular combination of errors and
attributes, and unless the software is prepared, chaos can
ensue.  In general, it is best to assume that the network is
filled with malevolent entities that will send in packets
designed to have the worst possible effect.  This assumption
will lead to suitable protective design, although the most
serious problems in the Internet have been caused by
un-envisaged mechanisms triggered by low-probability events;
mere human malice would never have taken so devious a course!
```

https://datatracker.ietf.org/doc/html/rfc1122

Balance interoperability with security – don't be **too** liberal in what you accept; a clear specification of how and when you will fail might be more appropriate

Colin Perkins | https://csperkins.org/ | Copyright © 2026 University of Glasgow

The Robustness Principle (Postel's Law)

"Be liberal in what you accept, and conservative in what you send"

Colin Perkins | https://csperkins.org/ | Copyright © 2026 University of Glasgow

The Robustness Principle (Postel's Law)

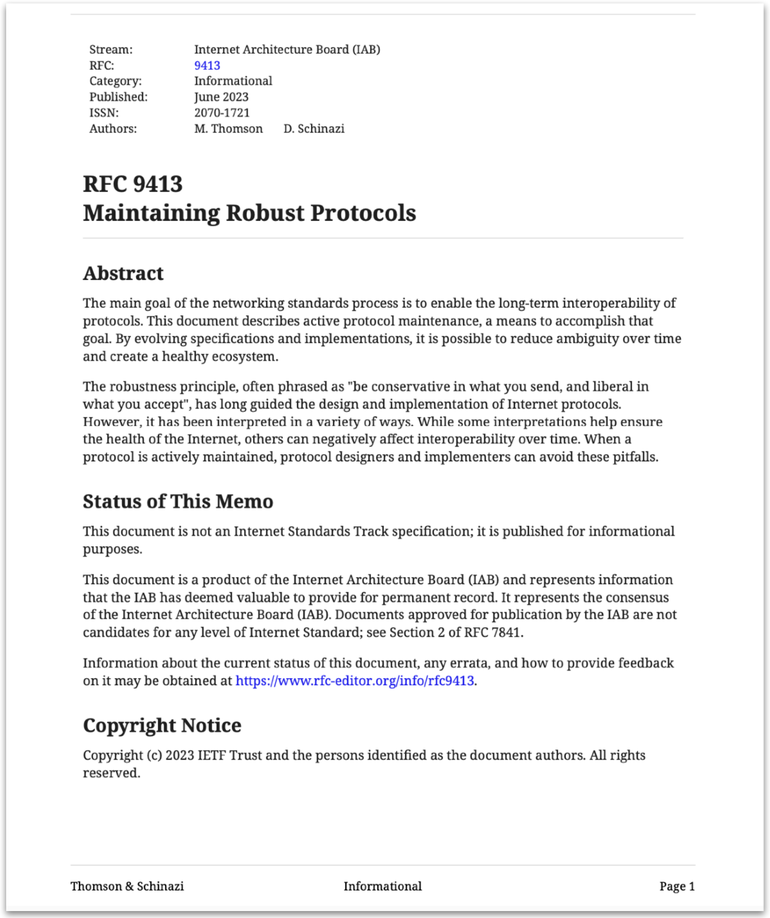
"Be liberal in what you accept, and conservative in what you send"

"Postel lived on a network with all his friends.
We live on a network with all our enemies.
Postel was wrong for todays internet."

– Poul-Henning Kamp

Colin Perkins | https://csperkins.org/ | Copyright © 2026 University of Glasgow

The Robustness Principle (Postel's Law)



Stream: Internet Architecture Board (IAB)
RFC: 9413
Category: Informational
Published: June 2023
ISSN: 2070-1721
Authors: M. Thomson D. Schinazi

RFC 9413
Maintaining Robust Protocols

Abstract

The main goal of the networking standards process is to enable the long-term interoperability of protocols. This document describes active protocol maintenance, a means to accomplish that goal. By evolving specifications and implementations, it is possible to reduce ambiguity over time and create a healthy ecosystem.

The robustness principle, often phrased as "be conservative in what you send, and liberal in what you accept", has long guided the design and implementation of Internet protocols. However, it has been interpreted in a variety of ways. While some interpretations help ensure the health of the Internet, others can negatively affect interoperability over time. When a protocol is actively maintained, protocol designers and implementers can avoid these pitfalls.

Status of This Memo

This document is not an Internet Standards Track specification; it is published for informational purposes.

This document is a product of the Internet Architecture Board (IAB) and represents information that the IAB has deemed valuable to provide for permanent record. It represents the consensus of the Internet Architecture Board (IAB). Documents approved for publication by the IAB are not candidates for any level of Internet Standard; see Section 2 of RFC 7841.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at https://www.rfc-editor.org/info/rfc9413.

Copyright Notice

Copyright (c) 2023 IETF Trust and the persons identified as the document authors. All rights reserved.

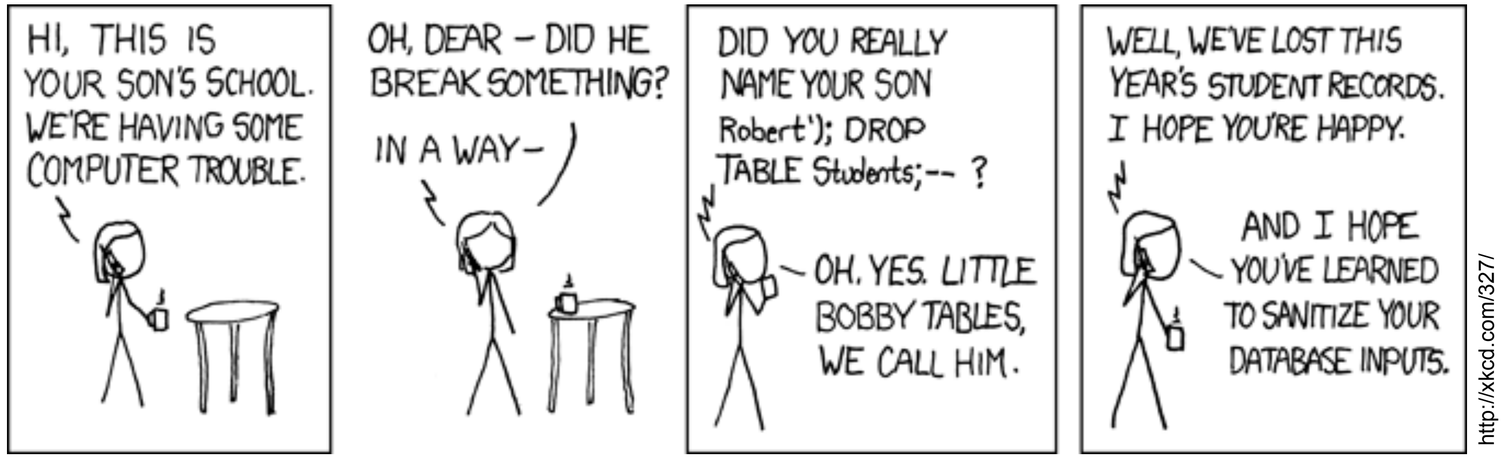
Thomson & Schinazi Informational Page 1

- Systems on the Internet are frequently subject to attack
- Systems must interoperate with other implementations that may be badly written
- Protocol implementations **must** have:
  - Robustness to software defects
  - Robustness to attacks
  - Robustness to the unexpected

- https://datatracker.ietf.org/doc/rfc9413/

Colin Perkins | https://csperkins.org/ | Copyright © 2026 University of Glasgow

Validating Input Data



http://xkcd.com/327/

- Networked applications work with data supplied by un-trusted third parties
- Data read from the network may not conform to the protocol specification
- Due to ignorance, bugs, malice, or a desire to disrupt services
- **Must carefully validate all data before use**

Colin Perkins | https://csperkins.org/ | Copyright © 2026 University of Glasgow

Writing Secure Code

- The network is hostile: any networked application is security critical
  - Must carefully specify behaviour with both correct and incorrect inputs
  - Must carefully validate inputs and handle errors
  - Must take additional care if using type- and memory-unsafe languages, such as C and C++, since these have additional failure modes
- **The best encryption doesn’t help if the endpoints can be compromised**

Colin Perkins | https://csperkins.org/ | Copyright © 2026 University of Glasgow

University of Glasgow

Secure Communications

- The need for secure communication
- Principles of secure communication
- TLS v1.3
- Discussion

Colin Perkins | https://csperkins.org/ | Copyright © 2026 University of Glasgow