Networking Secure Protocols

Protecting data in transit across insecure networks

Evolution of Transport Layer Security (TLS)

1 **Early 1990s**

Netscape recognizes need for secure web communication

2 **1995**

SSL 2.0 released as first public version

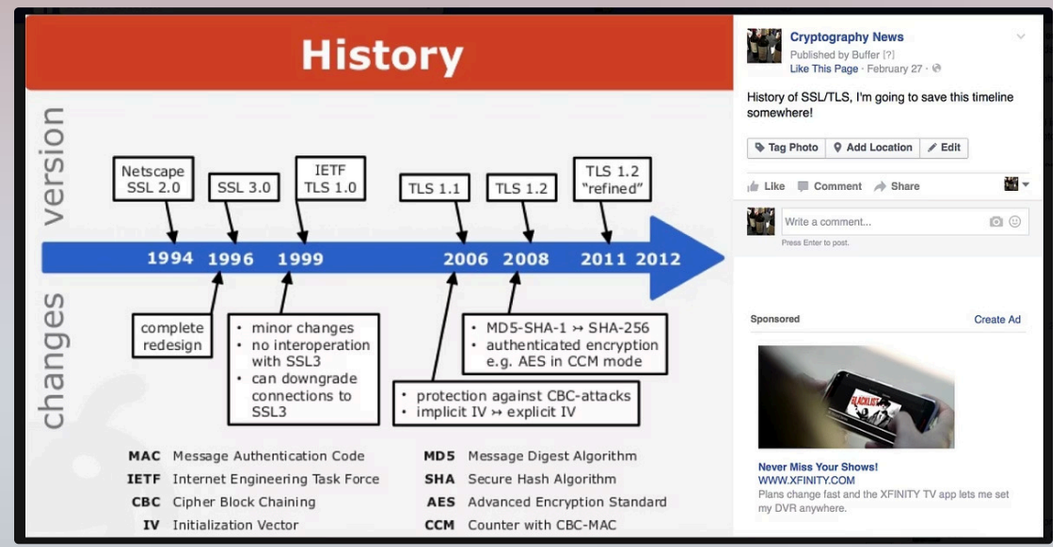
3 **1999**

IETF develops TLS 1.0 as upgrade to SSL 3.0

4 **2018**

TLS 1.3 released with significant protocol overhaul

TLS ensures confidentiality and integrity of data exchanged between client and server over insecure networks.



How TLS Works

Certificate Creation

Server administrator creates Certificate Signing Request (CSR)

Certificate Authority Verification

CA verifies and issues a digital certificate

Certificate Installation

Server uses certificate to identify itself to clients

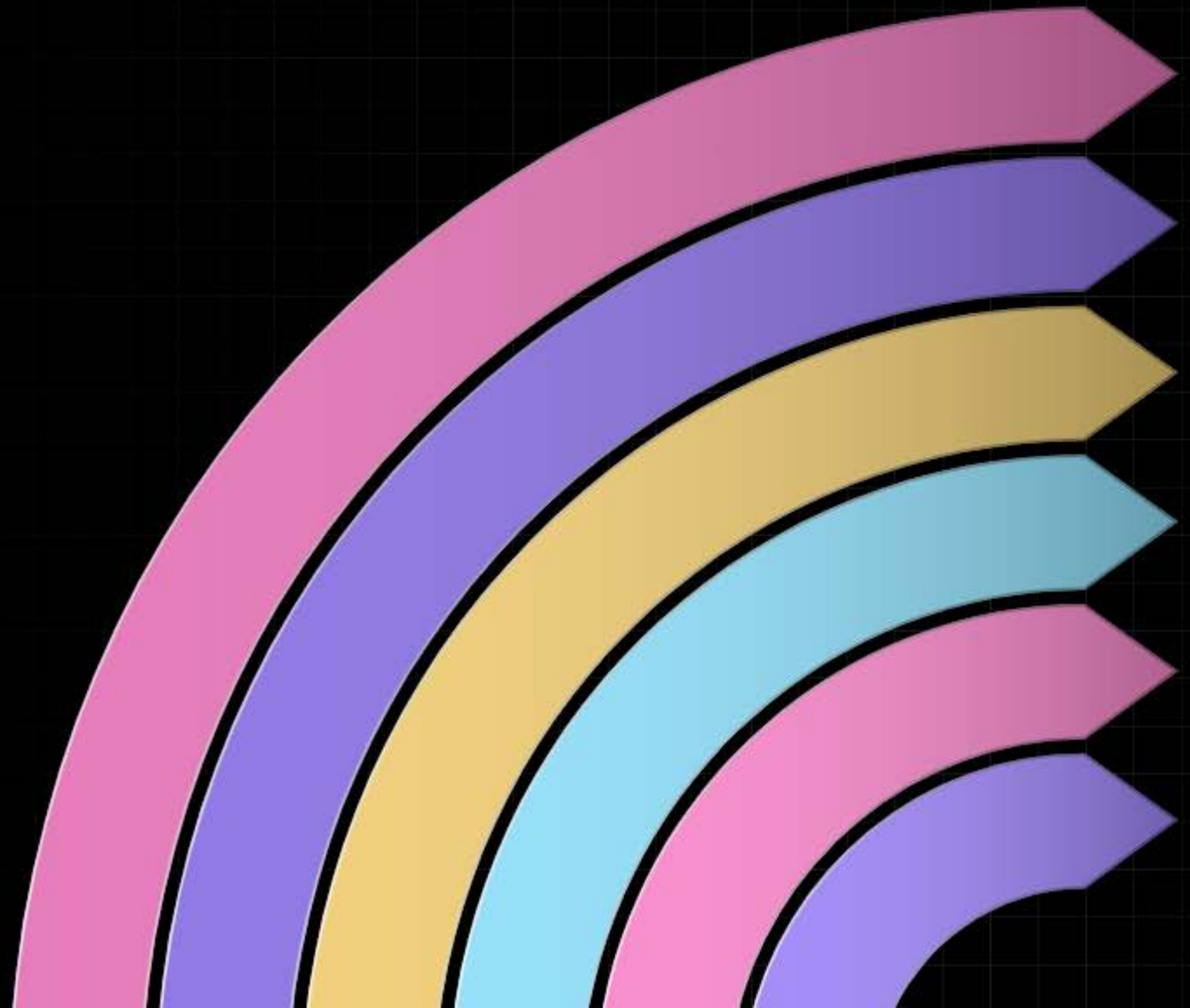
Client Validation

Clients confirm certificate validity using trusted CA certificates

Let's Encrypt provides free certificate signing, while self-signed certificates cannot prove server authenticity.

Understanding the Process of TLS Functionality

Exploring signed certificates, CSR, and trusted authorities in TLS



Importance of certificate validation

Certificate validation ensures the integrity and authenticity of secure connections.

Difference between trusted and self-signed certificates

Self-signed certificates lack external validation and are less secure than CA-signed ones.

Significance of trusted authorities

Trusted authorities are crucial for establishing a secure connection and preventing fraud.

Engagement of Certificate Authorities (CA)

CAs act as trusted entities that issue digital certificates for secure connections.

Creating a Certificate Signing Request (CSR)

A **CSR** is generated to request a digital certificate from a **CA**.

Role of signed TLS certificates

TLS certificates validate identities and ensure secure communication over networks.

HTTPS: HTTP Over TLS

HTTP (Insecure)

1. TCP three-way handshake
2. HTTP communication in cleartext

All traffic visible to anyone monitoring the network

HTTPS (Secure)

1. TCP three-way handshake
2. TLS session establishment
3. Encrypted HTTP communication

Traffic encrypted and unreadable without encryption key

Understanding HTTPS: Establishing a Secure Connection

A step-by-step guide to establishing a secure HTTPS connection using TLS

Initiate with a TCP three-way handshake.

This establishes a connection between the client and server.

Establish a TLS session for encryption.

TLS secures data transmission over the network.

Maintain a secure session until terminated.

HTTPS sessions remain secure throughout their duration.

Begin HTTP communication after securing the connection.

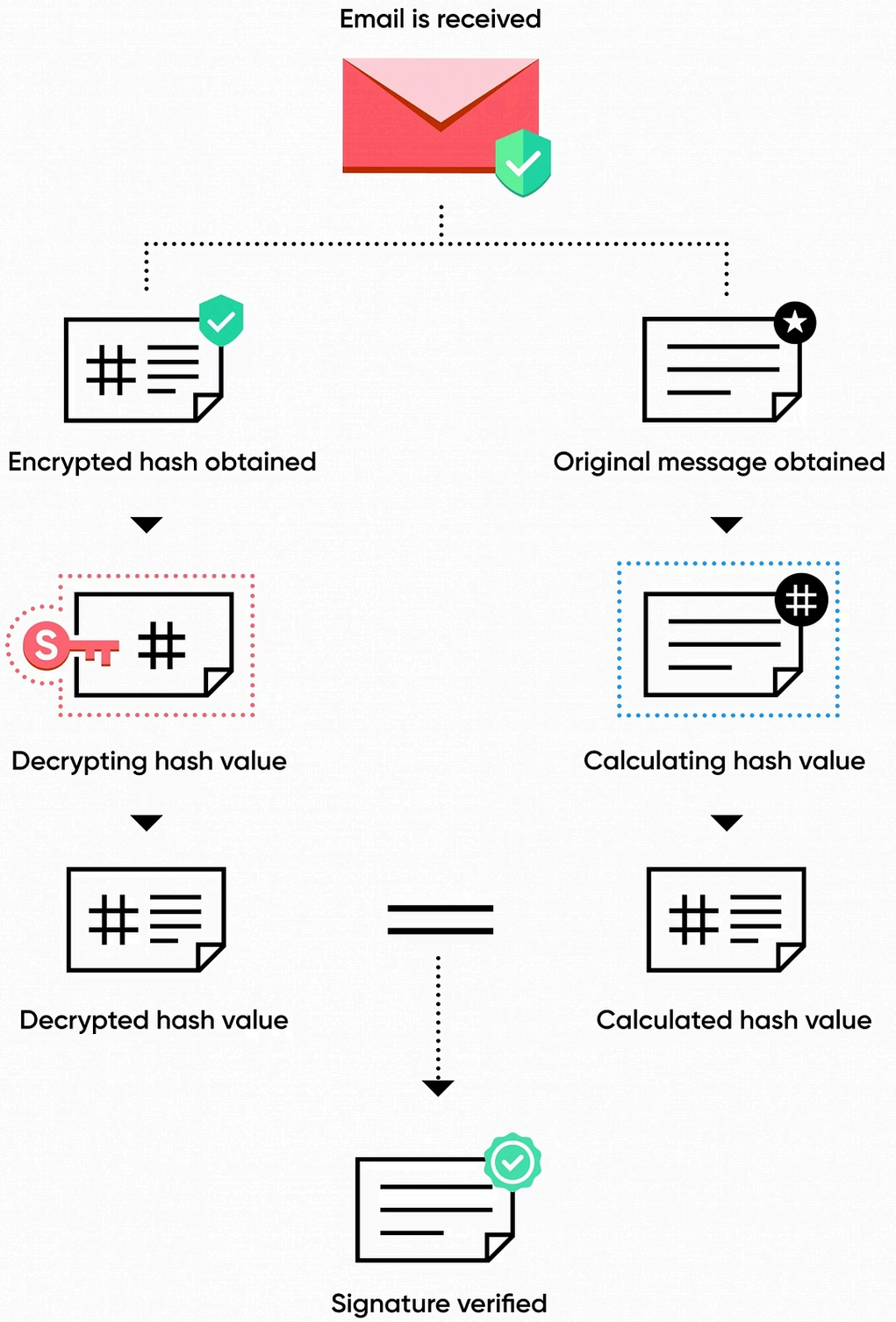
HTTP data is exchanged securely after establishing TLS.

Authenticate the server to the client.

Server certificates verify identity to prevent impersonation.

Ensure data integrity and confidentiality.

TLS provides encryption to protect data from eavesdroppers.



Email Protocols with TLS

SMTPS

Secure version of SMTP for sending email

Default ports: 465 and 587 (vs. port 25 for SMTP)

POP3S

Secure version of POP3 for retrieving email

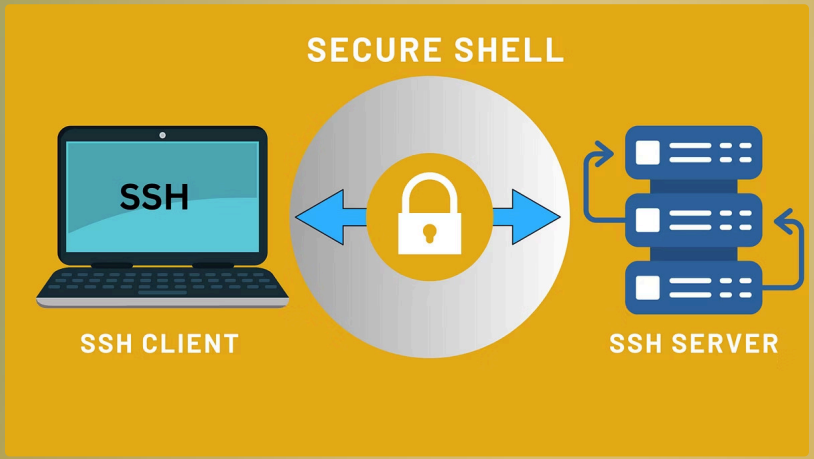
Default port: 995 (vs. port 110 for POP3)

IMAPS

Secure version of IMAP for accessing email

Default port: 993 (vs. port 143 for IMAP)

Adding TLS to email protocols provides the same security benefits as HTTPS.



Secure Shell (SSH)

History

- Developed by Tatu Ylönen in 1995
- SSH-2 defined in 1996
- OpenSSH released in 1999

Why SSH?

Replaced insecure TELNET protocol which sent all traffic including passwords in cleartext

Key Benefits of OpenSSH

Secure Authentication

Supports password, public key, and two-factor authentication

Confidentiality

End-to-end encryption protects against eavesdropping

Integrity

Cryptography ensures data cannot be modified in transit

Tunneling

Creates secure "tunnels" for routing other protocols

SSH also supports X11 Forwarding for running graphical applications over the network

SSH Usage

Basic Connection

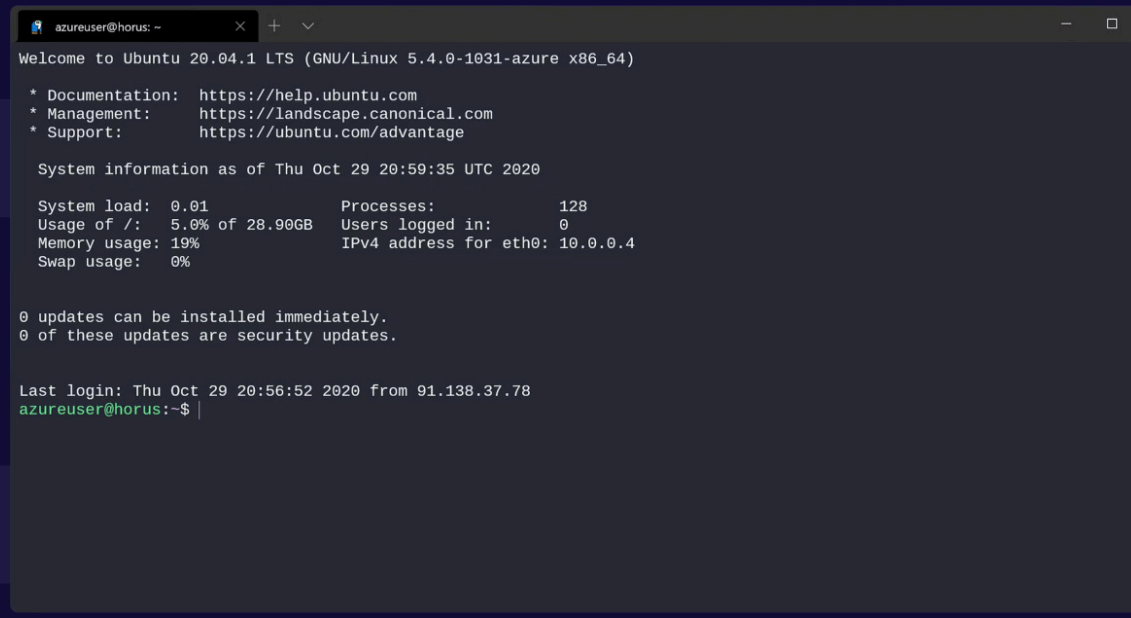
```
ssh username@hostname
```

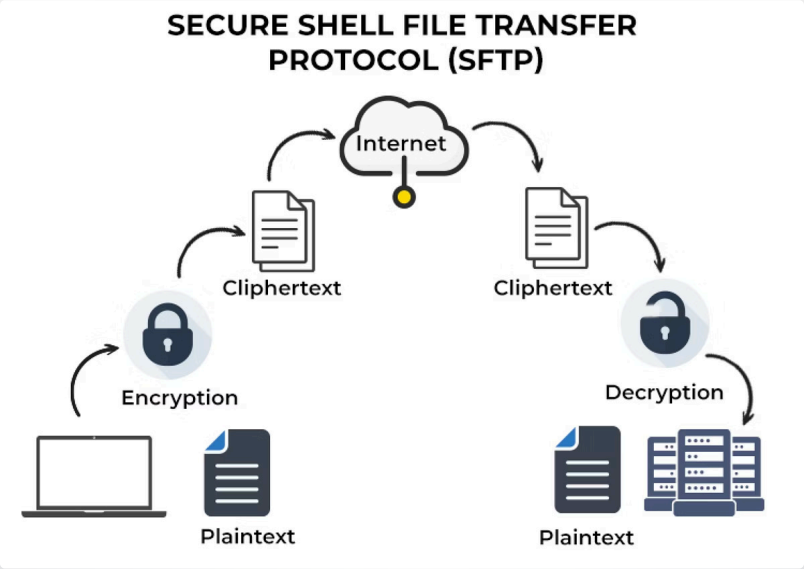
If username matches local user: `ssh hostname`

With X11 Forwarding

```
ssh hostname -X
```

Allows running graphical applications remotely





Secure File Transfer Protocols

SFTP

- SSH File Transfer Protocol
- Part of SSH protocol suite
- Uses port 22
- Easy to set up (enabled in OpenSSH)
- Uses Unix-like commands

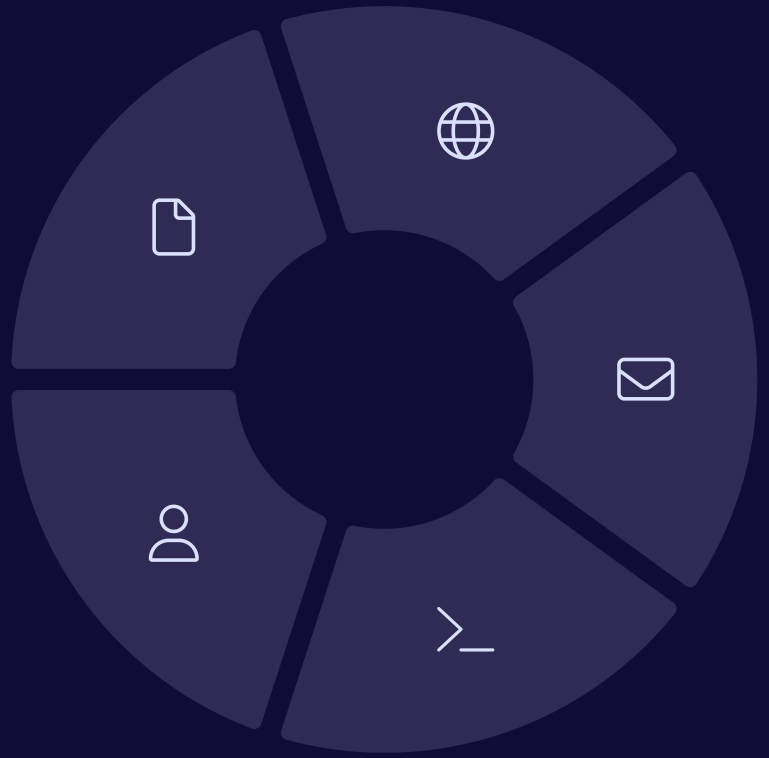
FTPS

- File Transfer Protocol Secure
- Secured using TLS
- Uses port 990 (vs. port 21 for FTP)
- Requires certificate setup
- Separate connections for control and data

Summary: Secure Protocol Ecosystem

TLS

Foundation for many secure protocols



HTTPS

Secure web browsing

Email Security

SMTPS, POP3S, IMAPS

SSH

Secure remote administration

File Transfer

SFTP and FTPS

These protocols enable our daily online activities with the security needed for sensitive operations like banking, shopping, and communication.