University of St Andrews

CS2003: Internet and the Web

Communication Security

Things you may have heard about …

University of St Andrews

What we will look at, briefly

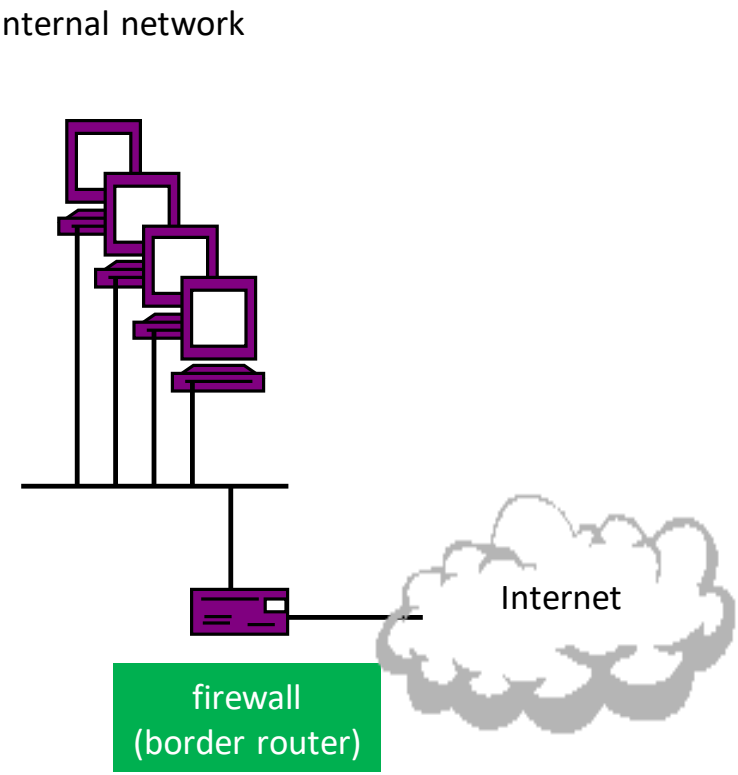
- Firewalls
- Cryptography:
  - "Encryption"
  - Secret Key
  - Public Key
  - Certificates
- TLS / SSL
- Some relevant laws
- **(BUT: there is much to security and privacy we just do not have time to cover!)**

No time to cover …

- Denial of Service (DoS)
- Botnets
- Malware
- Phishing
- Web-based attacks:
  - (SQL) injection
  - Cross Site Scripting
  - Man in the middle
  - Insecure encoding / decoding
- Many more …

Firewalls

- Prevents:
  - packets leaving site network.
  - packets entering site network.
- **A programmable packet filter:**
  - situated on border router.
  - implements filter **policy**.
- Instructions for filtering:
  - IP addresses, port numbers, protocols, other header fields, etc.
  - access control lists (ACLs): what is and is not allowed



Firewall applications

- Firewalls can run on routers
  - Traditional: acts as a “firewall” between traffic crossing networks
  - (firewall = a barrier within a building designed to prevent the spread of fire)
  - May be a performance bottleneck
  - e.g. iptables on Linux: network filtering
- Can also run on hosts
  - More computationally expensive: even more of a performance bottleneck
  - Different rules for different hosts
  - But sometimes easier for nomadic devices
  - e.g. NetGuard on Android: DNS filtering

Intrusion detection systems

- Firewalls use static rules
  - e.g. `iptables -A INPUT -p tcp -dport 22 -j ACCEPT`
  - allow input connections on port 22 (ssh)
- But you might want to update rules dynamically based on network traffic and activity
  - Intrusion detection system
  - Monitor network traffic at strategic points
  - Look for signatures of known attacks or patterns of suspicious activities (anomalies)
  - May need application-layer knowledge (deep packet inspection)
  - Raise alerts or update rules automatically
  - e.g. snort.org

Privacy and Security for Internet protocols

University of St Andrews

- IETF now considers security and privacy issues for all new protocol proposals, e.g.:
  - RFC3552 (BCP), July 2003, “Guidelines for Writing RFC Text on Security Considerations”.
- Snowden Leaks (June 2013):
  - https://www.bbc.co.uk/news/world-us-canada-23123964
  - Also see *CitizenFour* https://learningonscreen.ac.uk/ondemand/index.php/prog/08960EF2?bcast=115143289
  - RFC6973 (I), July 2013, “Privacy Considerations for Internet Protocols”.
  - RFC7258 (BCP), May 2014, “Pervasive Monitoring is an Attack”.

What does it mean to be “secure”?

Source - unknown

Security analysis

- Security **threats**:
  - what to protect against.
- Security **services**:
  - which (abstract) services will protect against the threats.
- Security **mechanisms**:
  - which algorithms and protocols to use to implement the security services chosen.
- Implementation:
  - the specific hardware, software and **policy** required.
- **General and application-specific requirements:**
  - a well-defined **security policy** is vital!

Privacy and Security

Example: making a purchase with a credit card

- Strong **security** for seller:
  - authentication of user
  - validation of ability to pay
- Strong **security** for buyer:
  - retailer is likely to be reputable
  - credit card company acts as “insurance” on purchase
- Very little privacy!
  - **buyer’s identity / location / payment / transaction details**
  - seller’s identity / location / transaction details

Privacy vs Security

- Privacy: control
  - e.g. who should have access to data
  - Might not always be the decision of the data subject (those to whom data pertain or who are identified by data)
  - A **huge** topic
    - Lots of different definitions of privacy
    - See Solove, *Taxonomy of Privacy*, or take CS5055 if you are interested!
- Security: implementation
  - what access is possible
  - i.e., how to do it

Cost / Benefit

- Security is not binary on/off.
- Security analysis helps you to establish the correct requirements for your needs.
- Possible performance drawbacks.
- You must balance:
  - operational needs
  - organisational structure
  - context of application
  - costs: implementation and operational
  - utility and usability
  - legal responsibilities
- (This all needs to go into a **site security policy**.)
- CS2003: threats **“on-the-wire”** vs “on-the-server”.

Aims of security

- Confidentiality
  - the concealment of information or resources
  - e.g., only sender and receiver should understand message content
- Integrity
  - the trustworthiness of information or resources
  - *data integrity*: ensuring the content of the information
  - *origin integrity*: ensuring the source of the information (authentication)
  - *non-repudiation*: ensuring that the creator of data cannot deny their existence, or that data were sent
  - e.g., sender and receiver can ensure that message has been sent and not been altered, or can confirm each other's identity
- Availability
  - the ability to use desired information or resources
  - e.g., services must be accessible and available to clients when the clients wish to use the service

Some common “on the wire” threats

University of St Andrews

- Disclosure
  - unauthorised access to information
  - e.g., traffic analysis
  - control plane and user plane
- Deception
  - acceptance of false data
  - e.g., man-in-the-middle (MitM), Trojan Horse
  - forged identities, fake servers, fake clients
- Disruption
  - interruption or prevention of correct operation
  - e.g., denial of service
- Usurpation
  - unauthorised control of some part of a system
  - e.g., message timing modification, forging of packets and data

“Making your system secure”

“Encryption software is expensive...so we just rearranged all the letters on your keyboard.”

Source - http://www.glasbergen.com/

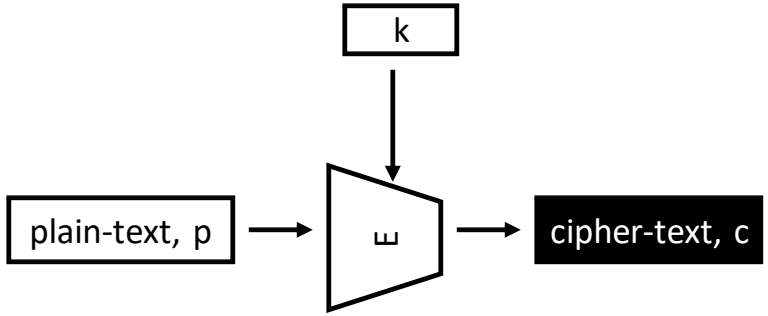
Some security mechanisms

- Authentication
  - certificates
  - security handshake
- Message authentication / integrity
  - digital signatures
  - message authentication codes
- Access control
  - access control lists
- Message synchronization
  - time stamps or unique message identification
- Line-loading
  - add dummy traffic

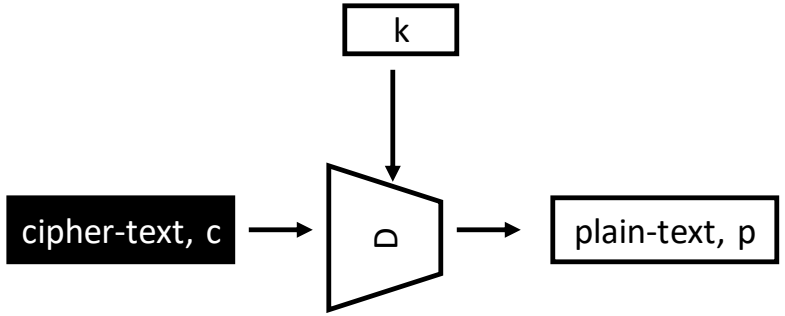
Many of these mechanisms rely on **cryptography**

Key-based cryptography

- Algorithm + key:
  - convert between **plain-text** and **cipher-text**
- Symmetric-key system:
  - **single key, $k_1 == k_2$**
  - used for both encryption and decryption
  - **must be known and kept secure by both parties**
- Public-key system:
  - **paired keys (matched keys)**
  - **k_1 and k_2 different but complementary**



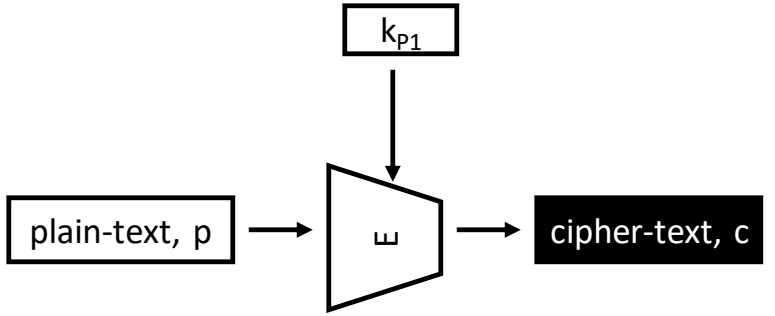
$c = E(p, k_1)$



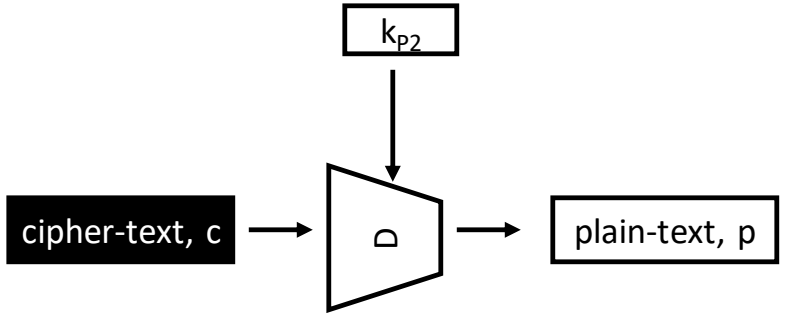
$p = D(c, k_2)$

Public-key systems

- Pair of matched keys:
  - one for encryption
  - one for decryption
- One key kept secret, k_{P1}
- One key made public, k_{P2}
- Not possible to deduce one key from knowledge of the other.
- Based on factorisation of large prime numbers:
  - due to Rivest, Shamir and Adleman (RSA)
- Harder to break than private key cryptography.
- More computationally expensive than private key cryptography.



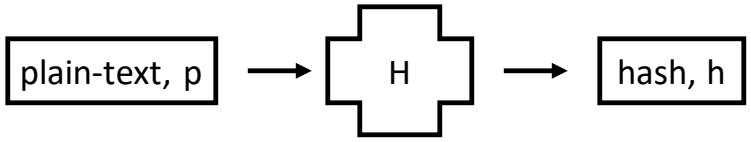
$$c = E(p, k_{P1})$$



$$p = D(c, k_{P2})$$

Hash Algorithms

- (Also known as digest algorithms.)
- Creates a fixed size bit pattern from any input of bits.
- A “strong checksum”.

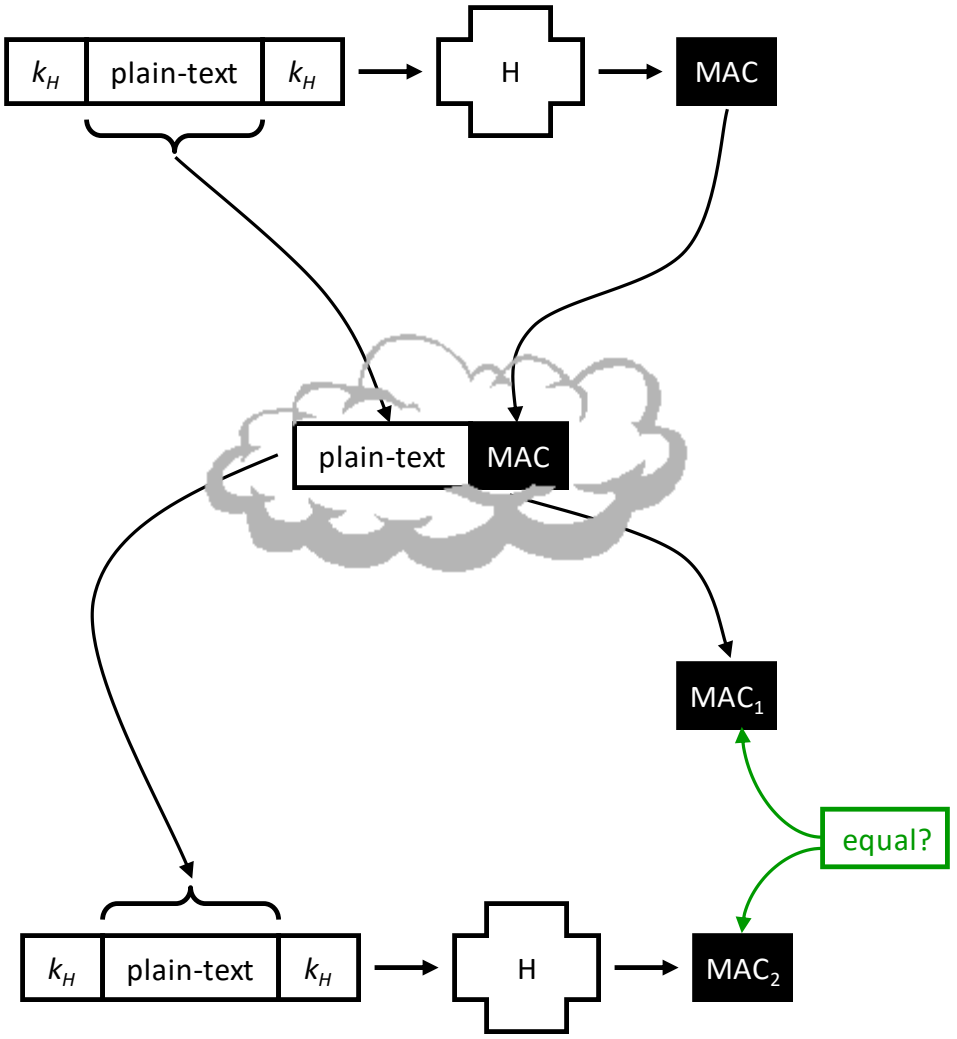


h = H(p)

- Properties:
  1. Fixed size hash value as output.
  2. Cannot reproduce original message from hash value.
  3. Very low probability of producing two messages with the same hash.
- Common usage:
  - SHA-256 (256 bits)
  - SHA-512 (512 bits)
  - deprecated
    - MD5 (128 bits)
    - SHA-1 (160 bits)

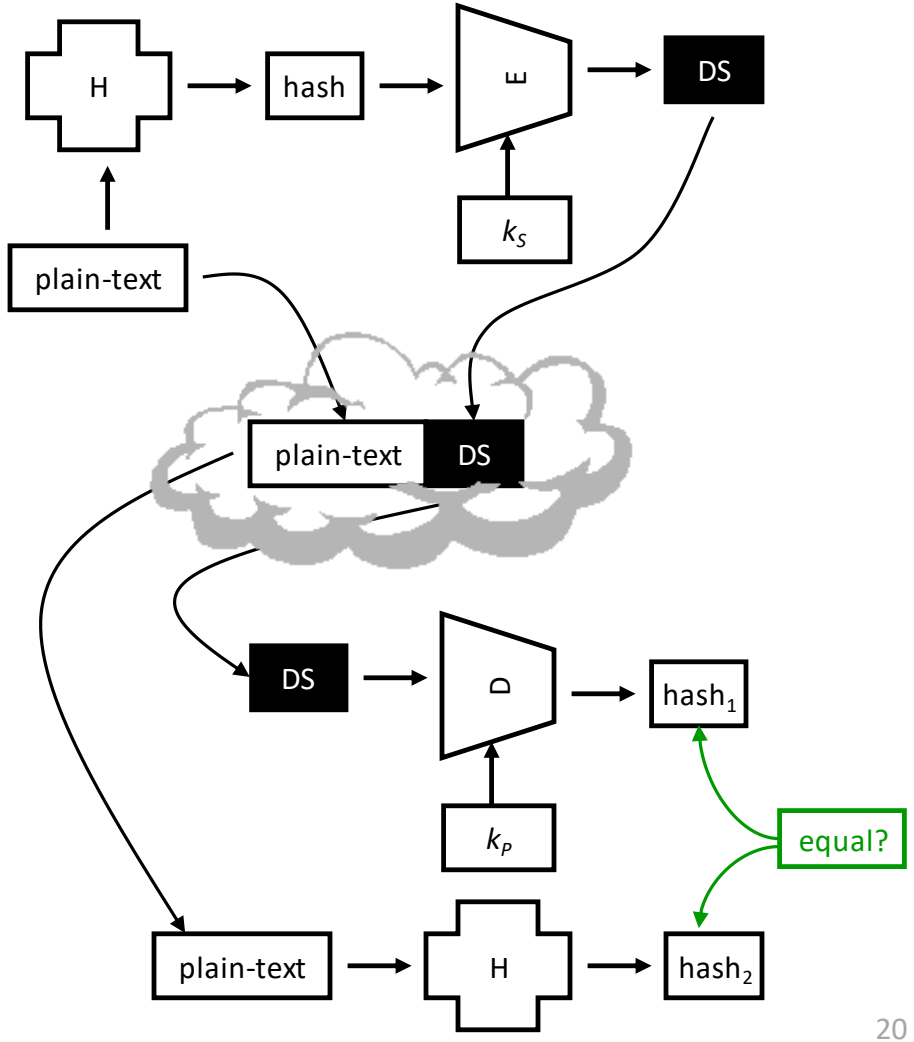
Message authentication codes

- Use of public key cryptography for DS:
  - computationally expensive.
- Use secret key, k_H:
  - "combine" plain-text with k_H as input to H.
  - sender and receiver must keep k_H secret.
- Authentication code:
  - MAC evaluated on k_H "combined" with message.
  - send message plus MAC.
  - receiver checks received MAC.



Digital signatures

- Digital signature is like a strong, secure checksum.
- Can be validated by the receiver of a message.
- Gives high assurance of message **authenticity** and **integrity**.
- Signature - public key:
  - create hash
  - encrypt with k_S
  - check at receiver using k_P



Before we continue…

https://www.bbc.co.uk/news/world-europe-55027641

A non-networked security example

University of St Andrews

- Back to our credit card example (buying a phone in a shop – pre lockdown!)

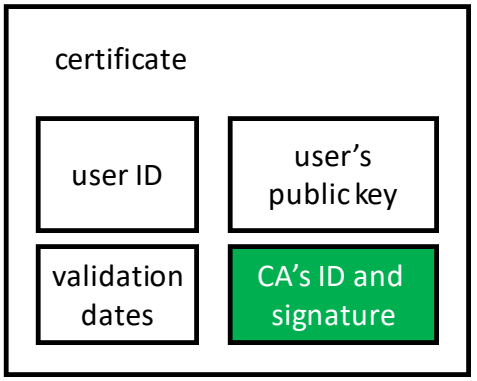
1. Choose phone — **begin** transaction
2. Ask shopkeeper to buy
3. Present credit card — **handshake**
4. Shopkeeper checks card
5. Shopkeeper verifies card
6. Shopkeeper checks PIN — **authentication** (shopkeeper does not trust you but trusts CC company)
7. Shopkeeper gives you the phone — **end** transaction

Who do you trust?

- Trusted third party?
  - who vouches for who?
  - how can you trust a certificate your receive?
- Certification Authority (CA):
  - issues a certificate.
  - CA **authenticates** the certificate using a **digital signature (DS)**.
- Anyone can check the DS on the certificate.
- CA keeps a list of revoked certificates.
- Registration Authority (RA) - identity of users.

Certificates

- Certificate contains:
  - user ID.
  - user's public key.
  - dates of validity.
  - CA's ID and signature.
- **Certification Authority**:
  - issues public key to user.
- On receiving a certificate, a user should:
  - check signature by using CA's public key.
  - check for revocation.
  - use directory service.
  - must trust CA!



- This is a simplified example.
- Real certificates contain additional information:
  - e.g. look at your own WWW browser for certificates.

Certification Authorities

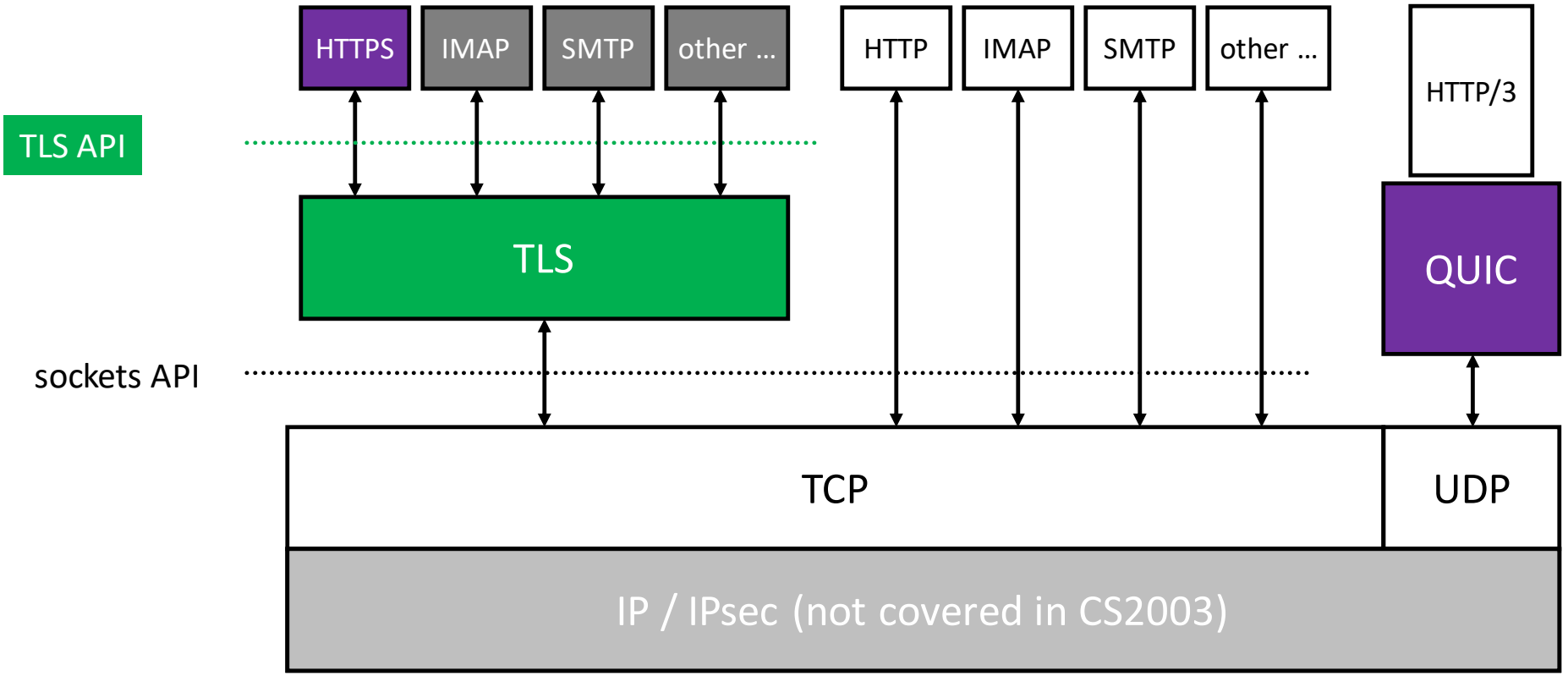
- Certificates:
  - provided by “well-known” CAs.
  - often come as part of the software (e.g. browsers).
  - new certificates can be installed
- Example CAs:
  - free: Let’s Encrypt
  - commercial: Quo Vadis, VeriSign, Thawte
- Can use self-signed (i.e., self-generated) certificates:
  - no trusted third party for verification of certificate.

Some common implementations

University of St Andrews

- Certificates:
  - public-key systems: e.g. X.509/X.511, PGP
- Message authentication/integrity:
  - SHA, keyed-MD5
- Private key encryption for privacy:
  - e.g. IDEA, AES, blowfish , 3DES
- TLS:
  - certificates, public-key based handshake, communication of session keys, end-to-end encryption

TLS (HTTP / TLS = HTTPS)



HTTPS = the new default web?

ANNOUNCEMENTS FIREFOX SECURITY

Firefox 83 introduces HTTPS-Only Mode

Christoph Kerschbaumer, Julian Gaibler, Arthur Edelstein and Thyla van der Merwe | **November 17, 2020**

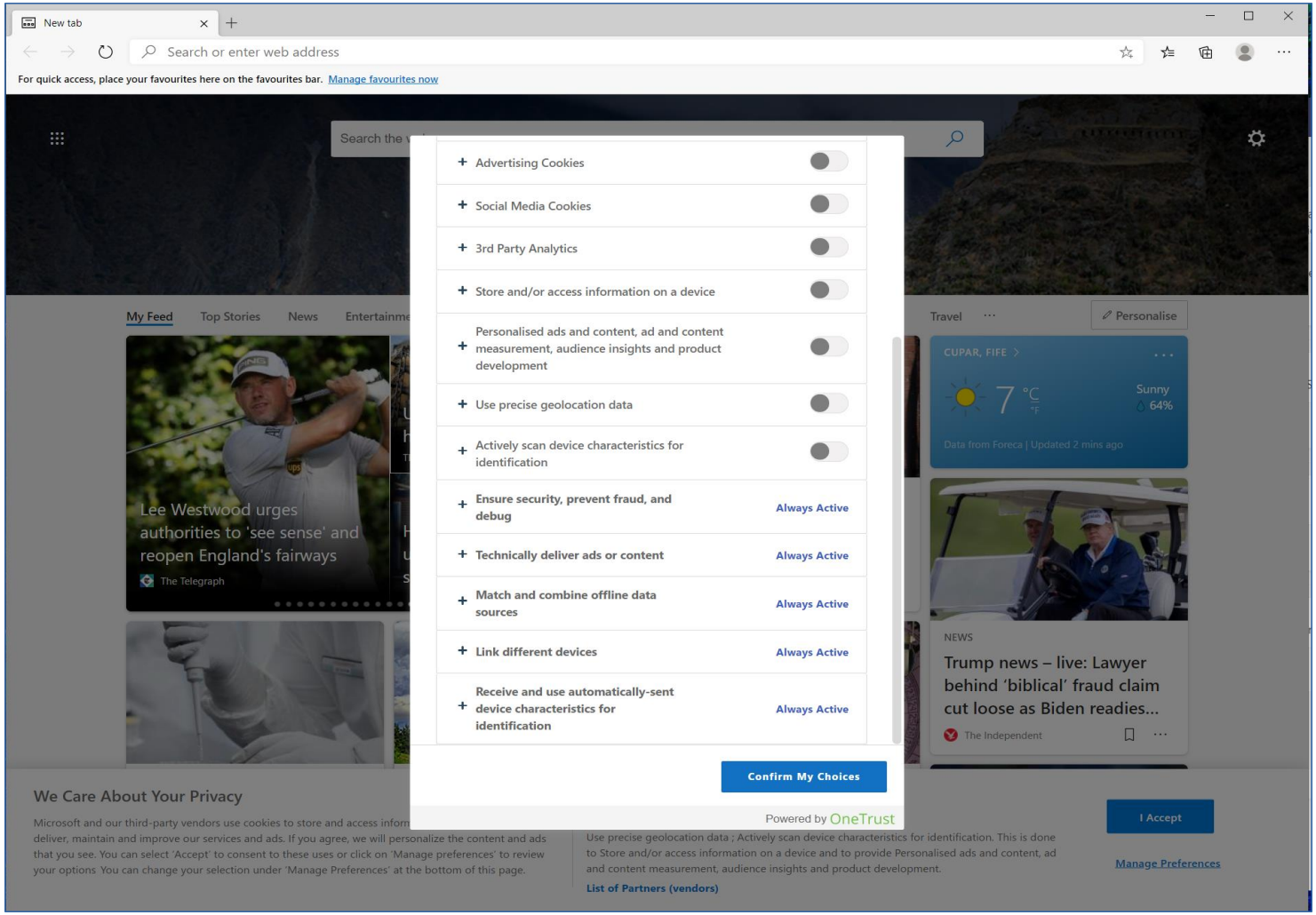
Security on the web matters. Whenever you connect to a web page and enter a password, a credit card number, or other sensitive information, you want to be sure that this information is kept secure. Whether you are writing a personal email or reading a page on a medical condition, you don't want that information leaked to eavesdroppers on the network who have no business prying into your personal communications.

That's why Mozilla is pleased to introduce HTTPS-Only Mode, a brand-new security feature available in Firefox 83. When you enable HTTPS-Only Mode:

- Firefox attempts to establish fully secure connections to every website, and
- Firefox asks for your permission before connecting to a website that doesn't support secure connections.

https://blog.mozilla.org/security/2020/11/17/firefox-83-introduces-https-only-mode/

A quick aside: rant about MS Edge



TLS (formerly SSL) (1)

- Transport Layer Security / Secure Sockets Layer
- Originally, SSL designed to provide secure HTTP sessions - HTTPS.
- Now a general API and can potentially be used by any protocol that operates over TCP.
- TLS/SSL is end-to-end:
  - sits above the normal sockets interface.
  - introduced as required into network without changes to network devices or network layer code.
- HTTPS = HTTP over TLS over TCP (HTTP/1.1 and /2)
  - HTTP/3 use QUIC, which has its own security.

TLS (formerly SSL) (2)

- API providing services that allow:
  1. a client to authenticate a server.
  2. (optionally) a server to authenticate a client.
  3. client and server to select crypto protocols for use over a communication session.
  4. client and server to exchange session keys securely.
  5. confidential sessions between client and server.
- Various crypto cipher suites permitting wide usage:
  - "very strong" (no export), "weak" (export OK.)
- Freely available implementations: OpenSSL, GnuTLS.

TLS example handshake

	client message	server message	description
1	ClientHello		TLS options proposed.
2		ServerHello	TLS options selected by server.
3		ServerKeyExchange	Certificate (including Public Key information).
4		ServerHelloDone	
5	ClientKeyExchange		Sends session key encrypted with server's public key. (Optional client authentication.)
6	ChangeCipherSpec		Activate new session state.
7	Finished		Secure session established.
8		ChangeCipherSpec	Activate new session state.
9		Finished	Secure session established.
			(encrypted communication session begins)

- Note that this happens **after** a TCP connection set-up:
  - High latency for HTTPS connection set-up in HTTP/1.1 and HTTP/2
  - ("International" version of TLS handshake has additional stages.)

Cipher suite examples

- Lots of different “cipher suites” available for clients and servers to negotiate
- Include: key exchange algorithm, encryption algorithm, MAC algorithm
- Examples:
  - Diffie-Hellman
  - AES secret-key encryption (256 bit key size).
  - SHA-256 MAC (256 bit)

TLS in Java

- Use SSLSocket instead of Socket
  - https://docs.oracle.com/en/java/javase/14/docs/api/java.base/javax/net/ssl/SSLSocket.html
- Minimal changes to application-layer code

```
import javax.net.ssl.*;
ServerSocketFactory ssf =
SSLServerSocketFactory.getDefault();
ServerSocket ss = ssf.createServerSocket(6666);
SSLSocket skt = (SSLSocket) factory.createSocket(host,
port);
skt.startHandshake();
skt.getSession();
```

Summary

- Firewalls/IDS
- Security:
  - Aims, threats, services, mechanisms
- Privacy and Security
- Cryptography:
  - Encryption
  - MAC
  - Certificates
- TLS
- Reading: Peterson & Davie Ch 8 or Kurose & Ross Ch 8 (both cover more than we have discussed)