Zusammenfassung Network and IoT Security - Roman Weber

1. Introduction

CIA Triad : Confidentiality, Integrity and Availability

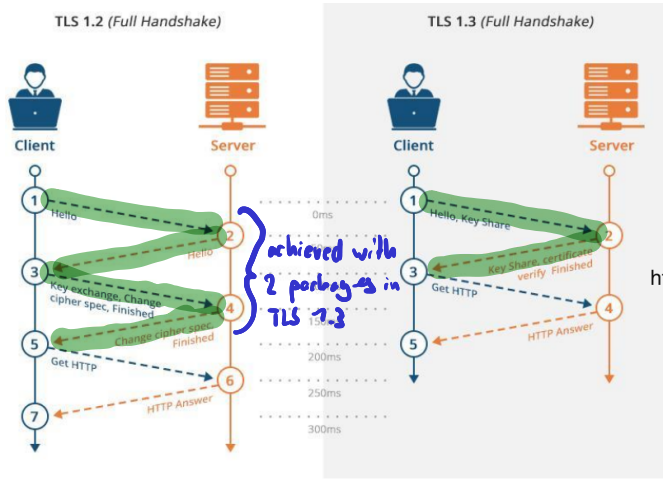
	Threat	Consequence	Countermeasure
Confidentiality	• Eavesdropping on the net • Theft of info from server • Theft of data from client • Info about network configuration • Info about which client talks to server	• Loss of information • Loss of privacy	Encryption, Web proxies
Integrity	• Modification of user data • Trojan horse browser • Modification of memory • Modification of message traffic in transit	• Loss of information • Compromise of machine • Vulnerability to all other threats	Cryptographic checksums
Availability / Denial of Service	• Killing of user threads • Flooding machine with bogus requests • Filling up disk or memory • Isolating machine by DNS attacks	• Disruptive • Annoying • Prevent user from getting work done	e.g. SCION
Authentication	• Impersonation of legitimate users • Data forgery	• Misrepresentation of user • Belief that false information is valid	Cryptographic techniques

TLS (Transport Layer Security)

- SSL is the old name and TLS 1.3 the newest version.

TLS handshake is quick and secure

- server and client must agree on enough information to have an encrypted conversation
- establish shared secret
- agree on ciphers and protocol
- authenticate server (and client, but rarely done)
- robust against tampering and attacks

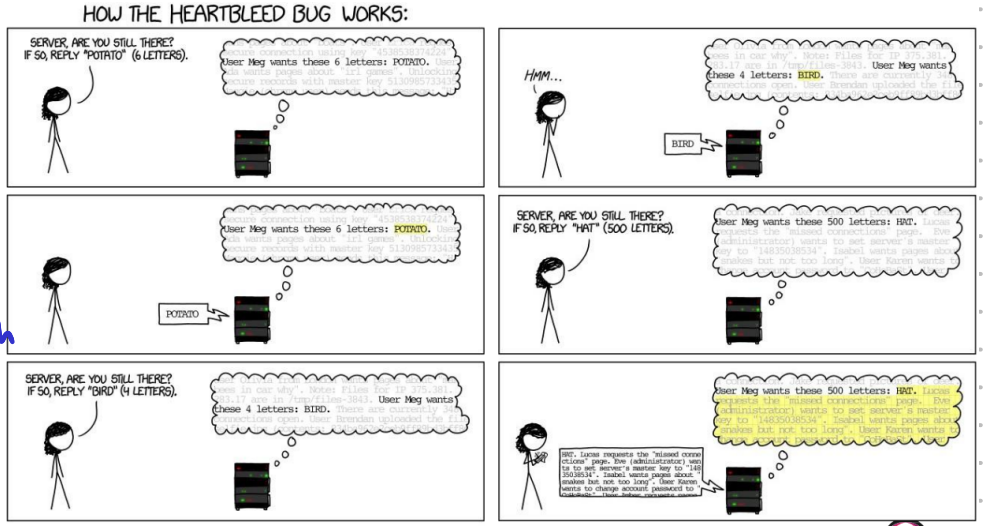


Heartbleed: (I think this is also covered in PETSec)

no, that is Heartbleed

- Heartbeat is a TLS connection keep-alive feature
- Heartbleed is an implementation flaw in the extension
  - ↳ essentially you lie about the payload size

There is a XKCD about it ->

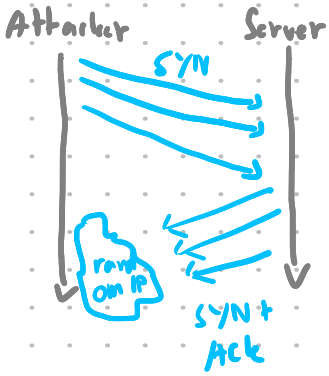


To fix:

- update openssl
- The openSSL implementation now checks for payload length

SYN - Flooding attack

Process control Block (stores info about connections)



Idea: Fill up the PCB queue with half open TLS packages so it can't taken any more connections

Countermeasures:
- SYN cookies -> calculated from info in package
- Server does not store half open connections
- if client is an attacker -> SYN cookie does not reach him (random IP)
- if client is no attacker, replies with SYN cookie +1 (sequence number)

TCP Reset attack:

- Reset Flag immediately breaks connection

Goal: break up TCP connection between A and B

Done by spoofing Reset package

TLS

FREAK (TLS downgrade) Factoring RSA Export Keys

- (Old) TLS support weak cipher suites
- Goal is to agree on a weak cipher that can be broken easily (and cheap)
- Man-in-the-Middle used to downgrade TLS cipher

elliptic curve ↓ deterministic random bit generator ↗

Dual EC DRBG / cryptographic backdoors

CSPRNG: cryptographically secure pseudorandom number generator

- not all random number generators are suitable for encryption

- Might have a NSA backdoor.
- NOBUS: nobody but us is essentially a locked backdoor
- constant allows prediction of output -> eg. TLS keys

counter measure: nothing-up-my-sleeve numbers (all constants generated in explainable way)

-> Exercise not added yet

2. Cryptography

Goal: secure channel =
- confidential, can not eavesdrop or learn anything about msg
- authenticity, cannot tamper with message

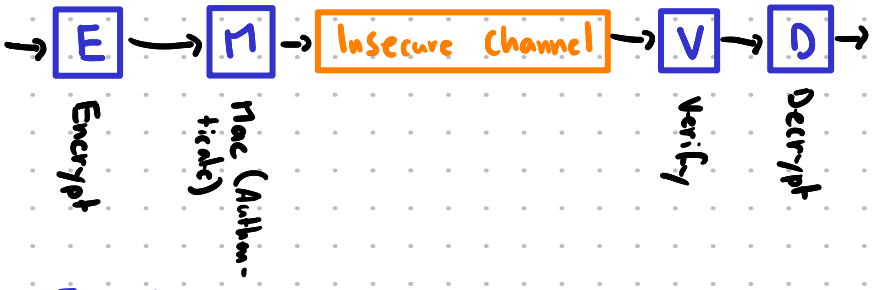
Starting point

Symmetric-key cryptography

Prerequisite: Alice and Bob have a shared secret key. This is not trivial

Achieve secure channel

1. achieve authentic channel
2. add confidentiality

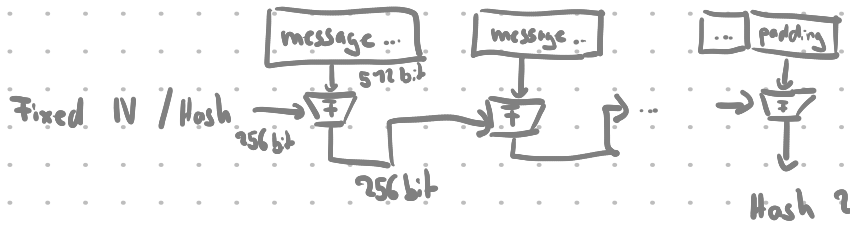


MAC (Message Authentication Code) Eg. SHA3

Idea: send m || H(m) → this can be computed by anyone

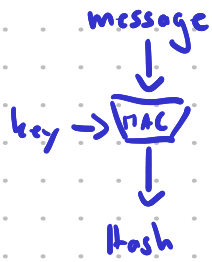
Thus MAC is essentially a keyed Hash function → needs to be
- oneway
- collision resistant

Merkle-Damgård (MD5, SHA1, SHA2 → all insecure)



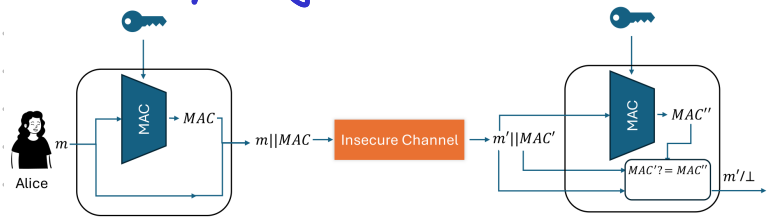
! length-extension attack possible !

HMAC SHA-256



also vulnerable to length extension attacks

Authenticity using MAC



- Compute MAC(m)
- Send m||MAC(m)

- Compute MAC(m')
- MAC(m') ?= MAC'
  - Output m' if yes
  - Output ⊥ if not

Difference Integrity and Authenticity

- ensure that a message has not been altered
- Does error correction of transit errors

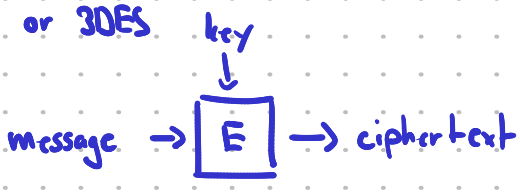
- ensures that the message has not been altered since Alice sent it
- makes sure the message is from Alice

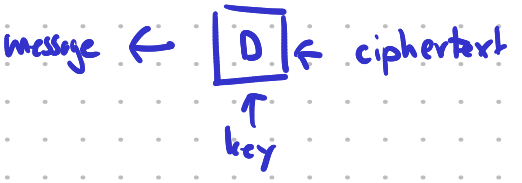
Block Cipher

AES (de facto standard) 128 bit or 3DES 64 bit

For messages of a fixed length

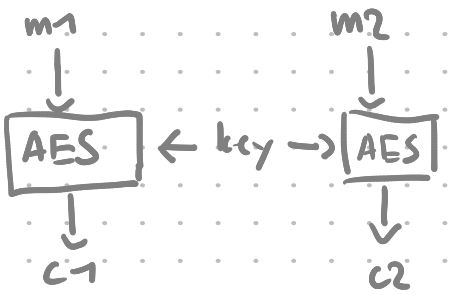
- padding used to make size work





ECB (Electronic CodeBook) don't use

if m1=m2 => c1=c2 -> repeating patterns are visible

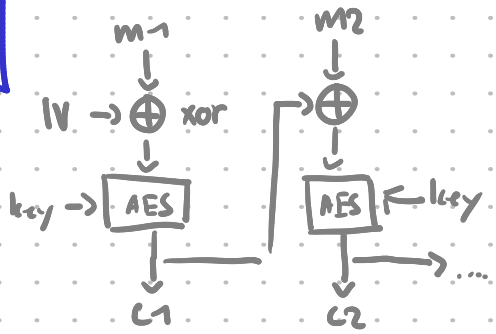


Block Cipher modes

galois counter mode

or GCM

CBC (continious block chaining)



- better to use this method
- This would allow to change c2 and m2 will decrypt differently, to authenticate the message

Sweet32 attack:

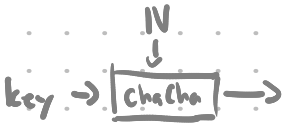
For these block ciphers it is expected to have a collision after $2^{n/2}$

With 3DES being 64 bit => collision after ~2^{32} which we can expect to happen

Slidedeck 2 P.26 look at graphic again

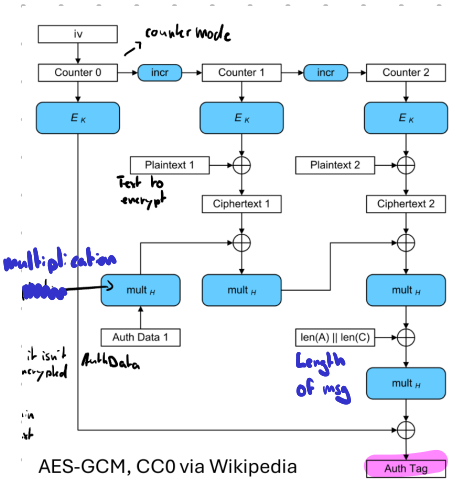
Stream cipher ChaCha

- generate pseudo random key stream
  - never ending stream of keys
- XOR encrypt with key stream



AEAD Authenticated Encryption with associated data

- eg: AES-GCM, ChaCha20-Poly1305



AES-GCM, CC0 via Wikipedia

GCM will create the authtag, against which one can check if any of the ciphertext has been altered

Done for basically every package you send

Computerphile video

https://www.youtube.com/watch?v=-fpVv_T4xwA

3) Encryption 2

Use standards and don't create your own algorithms

Establishing a shared key

so Alice and Bob both get the key, but Charlie doesn't

Good practice: ephemeral keys (kurzlebig), randomly generated for each session and short lived

Private key: sign or decrypt message
Public key: verify or encrypt message
} RSA or ECDSA (Digital Signature algorithm)

Perfect Forward Secrecy: attacker with access to longterm PIC can't decrypt -> not given because DH is breakable with Quantum computers (asymmetric)

Slide goes over DH, but I don't expect that we need to know this for the exam

BB84 Quantum Key Exchange

Idea: cannot eavesdrop without changing the data.

- Needs authenticated channel -> reading the photons changes them
- Needs direct connection like fibre optics

BB84 - Example

Example without interference

Alice bit	1	1	0	0	0
Alice base	+	x	+	+	x
Alice sends	| (=1)	/ (=1)	- (=0)	- (=0)	\ (=0)
Bob's filter	+	+	x	+	x
Bob's result	| (=1)	| or -	/ or \	- (=0)	\ (=0)

Example with eavesdropping

Alice sends	|	/	-	-	\
Eve's filter	x	any	any	+	+
Eve's gets & sends	/ or \			-	- or +
Bob's filter	+	+	x	+	x
Bob's result	| or -	| or -	/ or \	-	/ or \

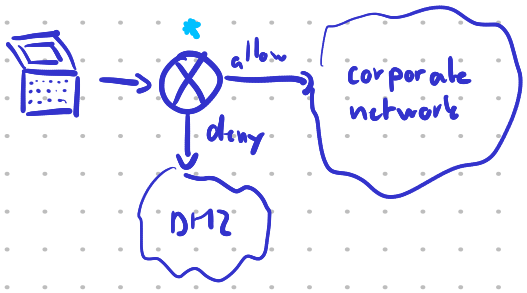
Haven't fully understood this slide ↑

Notes from a video

1) Alice random photons with base
2) Bob measures these photons with a random base
   -> as there are two bases, ~50% of photons must be measured correctly
3) Alice shares what base she used (but not the bits/values)
4) Bob now knows which photons he measured correctly -> key
5) an error in the key indicates eavesdropping

4) NAC (Network access control)

What is allowed to access the network - basically a bouncer.



* can be multiple things

Layer 2: switches / access points (IEEE 802.1x), VLAN

Layer 3: VPN

- Firewalls, DHCP management, ...

IEEE 802.1x using EAP

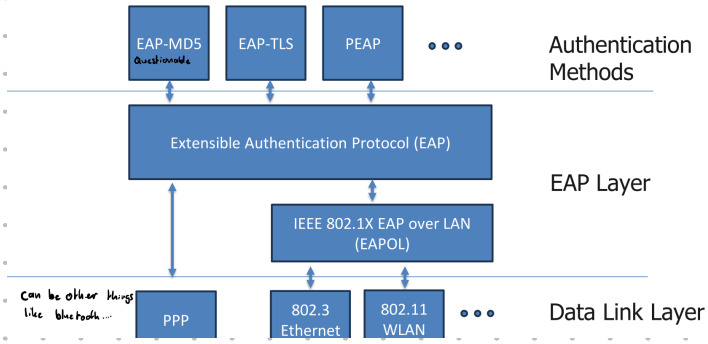
(802.1x: name of standard; EAP: extensible authentication protocol)

802.1X Supplicant → Laptop / Device

802.1X Authenticator → Switch / AP (doesn't do any authentication)

802.1X authentication server → handles authentication and authorization (Radius)

EAPol_Start message to start authentication (normally done by supplicant (or authenticator))



EAP-MD5: outdated

EAP-TLS: X509 certificate used for authentication (no pw)

PEAP: protected EAP using X.509 cert for TLS tunnel between supplicant and server

↳ within TLS, MSCHAPv2 for credential transmission

↳ outer identity normally anonymous

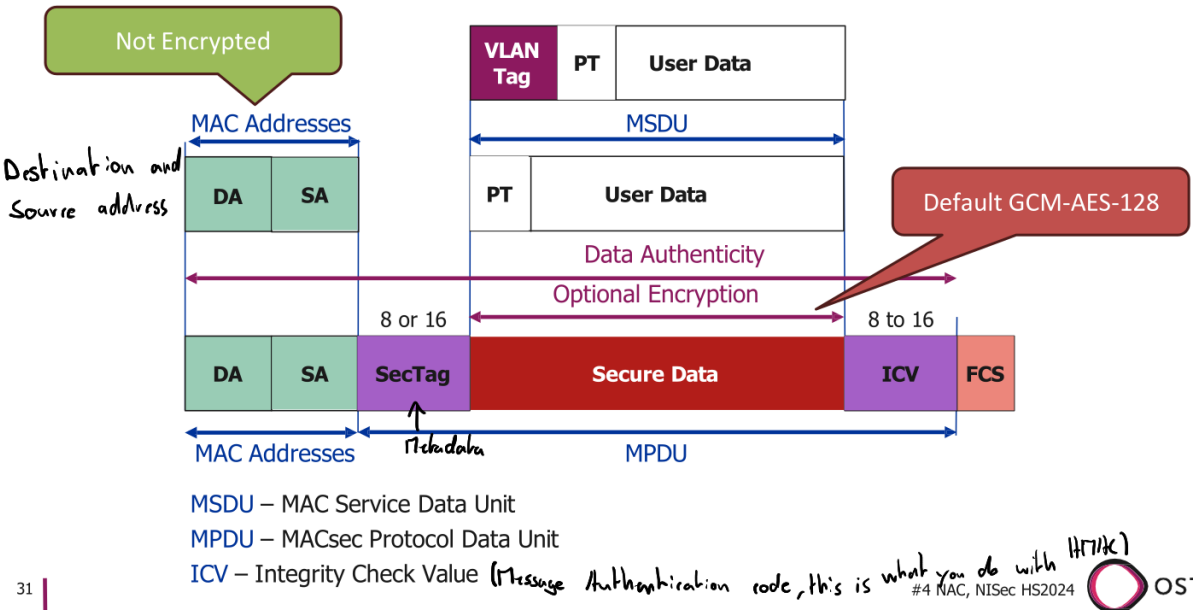
EAP master session must be shared with authenticator that is established here, otherwise he has no idea about the connection

DevID is the secure device identifier, based on hardware. Hardware is more difficult to change without anyone noticing. DevID used for device authentication.

MACsec IEEE 802.1AE

Operates at medium access control layer (so 2) and defines connectionless data confidentiality and integrity for media access independent protocols.

IEEE 802.1AE MACsec Frame Format



MSDU – MAC Service Data Unit
MPDU – MACsec Protocol Data Unit
ICV – Integrity Check Value (Message Authentication code, this is what you do with HMAC)

Secure channel: ?

Secure association: ?

not quite sure

MACsec key agreement (IEEE 802.1x)

master key → connectivity association key [CAK] → [SAK] secure association key

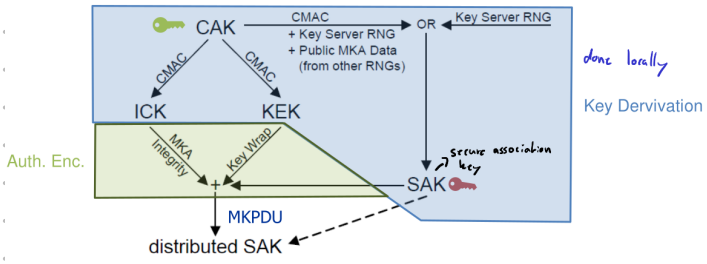
KDF: key derivation function

- HMAC or CMAC
- deterministic
- creates fresh keys

Todo: purpose of these keys

sometimes: KDF(key, label, context, length) → output
↳ to generate multiple keys using the same context

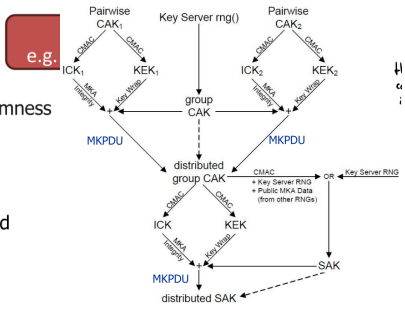
MKA distributes random SAK using CAK



MKPDU – MACsec Key Agreement Protocol Data Unit – carried via EAPOL
CAK – Connectivity Association Key – pairwise or group root key
ICK – ICV Key – used for MKPDU Data Integrity
KEK – Key Encrypting Key – used for AES Key Wrap in MKPDU
SAK – Secure Association Key

1. Members of CA have shared CAK
2. Key Server (KS) generates SAK
   - From CAK via KDF or using fresh randomness
3. KS distributes SAK to other members of CA
   - Using MKPDU over EAPOL
   - Via "ad-hoc" secure channel
     - Requires keys for integrity (ICK) and encryption (KEK)
       - Generated from CAK via KDF

e.g.



Another way to enforce NAC is to assess if a device is healthy. This can be done with multiple tools or programs. A TPM chip can help too. Similar to intune.

TPM: Chip on each system that can wrap keys in order for them to be stored securly on a disk. Disk can't be moved to another device to decrypt keys.

5 DNSSEC DNS Security Extension

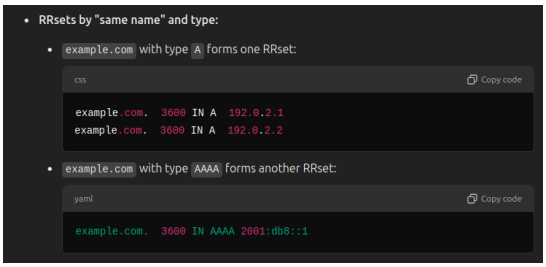
Start of the lecture is about DNS, but I'm going to skip that.

DNS responses are not signed, which is why the response might not be genuine
↳ DNS cache poisening

DNSSEC provides End-to-End protection using signatures (not very widely in use though)

all records with the same type and name grouped into record set (RRSet)

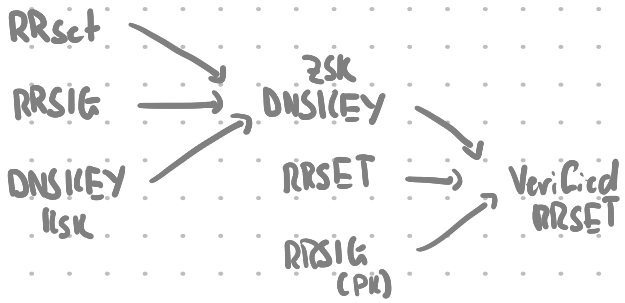
RRSETs are digitally signed (not records)



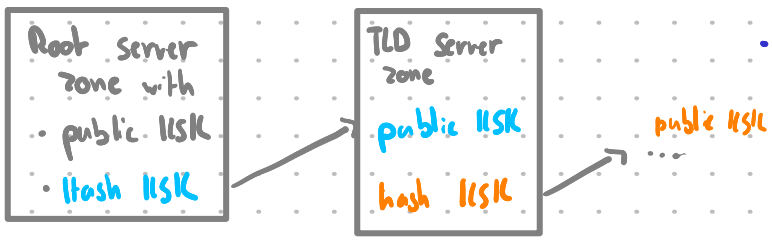
Each zone has a key pair (zone signing)

- private key to sign RRSETs
- public key published in RRSIG record

Then there is a key signing key which verifies the key the same way it works for the zone signing record)



Similar to the TLS chain of trust, there is a DNSSEC chain of trust.



- To check validity, hashes KSK and checks it against the parent zone.
- If the match → trust the chain

Denial of existance

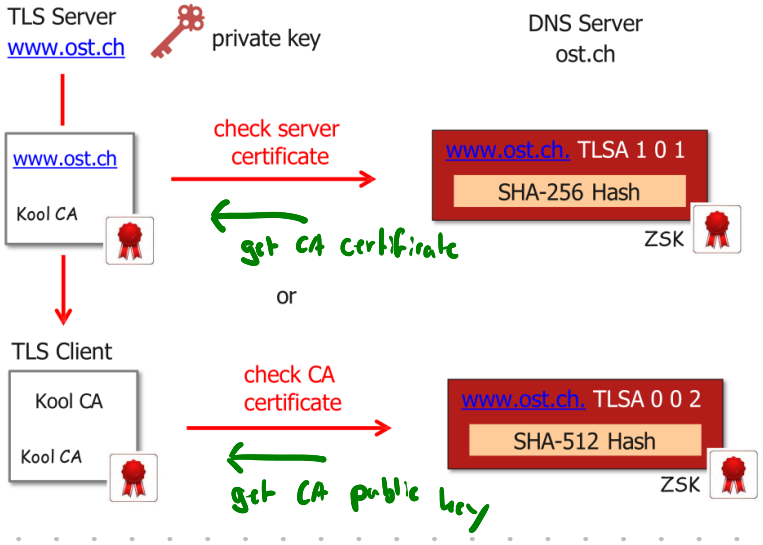
Normal DNS returns an empty record if it doesn't exist

NSEC (Next Secure) gives you back the next valid record and basically says the requ-
(or NSEC3) -sted one doesn't exist.

→ Does not allow straight enumeration of zone data (how?)

DANE DNS-Based authentication of named entities

DANE is a protocol that allows X509 certificates to be bound to DNS records using DNSSEC.



Step two

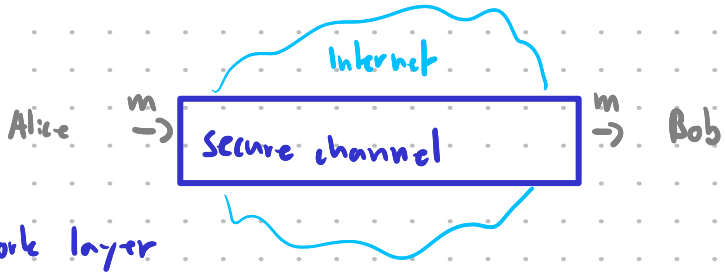
ICANN Key ceremonies: This is where the root key is used / rotated. Access is split across multiple people and the very formal process is filmed and streamed on youtube

To make sure everybody trusts this key

last one is 4 hours long

6) IPSec

IPSEC in a nutshell:
kind of like a VPN connection



IPSEC operates on the network layer

AH → authentication Header: • Protects integrity of payload and parts of the IP header
• Does not prove confidentiality

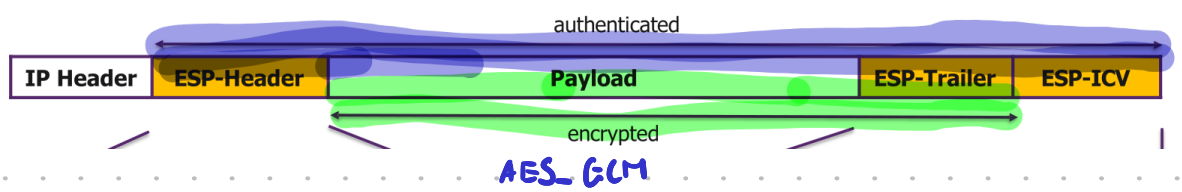
parts of AH:

- Next Header: Protocol of payload (UDP, TCP)
- Security parameters Index: authentication algorithms
- Sequence Number Field: prevent replay attacks*
- ICV (Integrity check value): HMAC or CMAC to prove integrity

* Sender increments this number with every package. There is a anti-replay window defined, in that [ ] packets are allowed. Needed as packages don't arrive in order.

Encapsulating Security Payload (ESP)

• IP header not protected



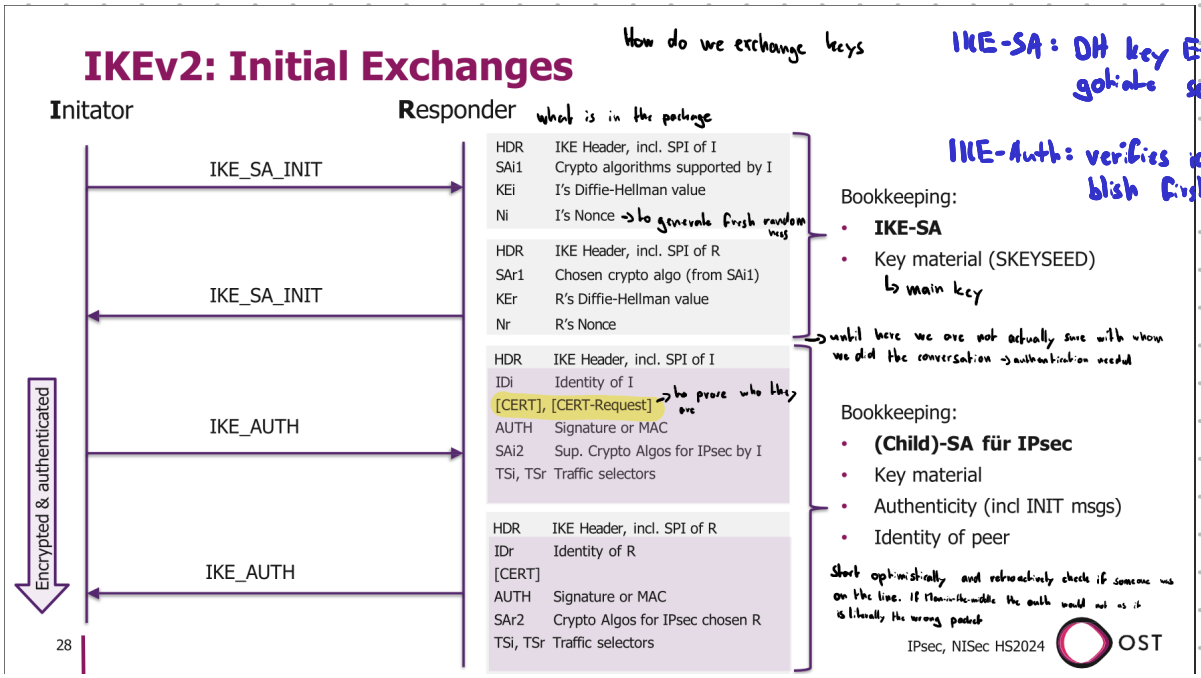
AES_GCM

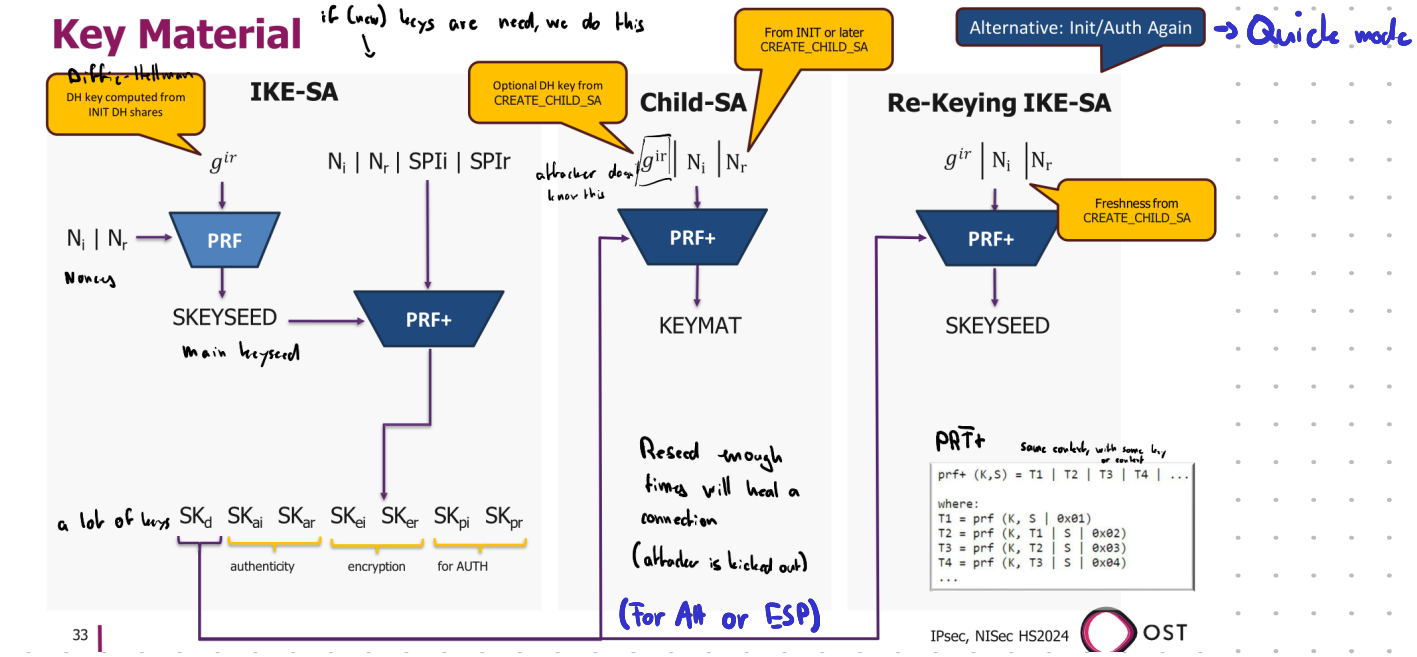
Modes of operation: · Transport: Protects IP packets directly
· Tunneled: Wraps IP packet into a new IP packet
↳ Site-to-Site VPN

SA Security Association to agree on things like Cipher suite, Protocol (AH or ESP), mode of operation...

IKEv2 Internet Key Exchange

Protocol for mutual authentication and build a security association





This allows to create new keys

IPSEC & NAT: To make NAT work, another header is added

Other VPN solutions: SSL VPN (OpenVPN) based on TLS, Layer 2 Tunneling protocol

7) Firewalls

Firewall originally is literally a wall against fire.

Types of Firewalls:

- Application Layer:
  - Deep packet inspection
  - Application gateways
  - Termination of connection and inspection
- Transport Layer:
  - check header
- Network Layer:
  - primitive packet filtering

Anything modern can do all of those things

FW attacks and countermeasures

- IP address spoofing: discard packages with an inside source IP
- Source routing attack (specifies route across internet in hopes of bypassing security checks): discard pkg using this option
- Tiny fragment attacks (extremly small fragments so TCP header is split): define minimum packet size

Circuit Level Gateway:
- Does not permit end-to-end TCP connections but sets up two TCP connections → relays them
- Typical use case if you trust internal users

Next Gen Firewall:
- fairly outdated too
- performs more in-depth packet inspection
- Does VPN (lol)

DMZ is mentioned, but nothing I don't know already

AIR Gap:
- something physically needs to move from network A to B as there isn't a connection between them.
- Very secure as physical access is needed → not invulnerable (eg bad USB stick)

8) IDS Intrusion Detection System

Firewall checks what can get in, the IDS (2nd line of defence) tries to uncover if somebody is already in

Attacker POV: using sequence of commands to gain more privileges

Goals of IDS: Detect wide range of attacks (known and unknown), detect in timely fashion, present analysis simple, Be accurate

IDS components and terminology:

Sensor: hard- or software that monitors client OS or network traffic

Event: noteworthy event

Alarm: Event gone bad → action to happen (human or automated)

DBS: log collected and events/alerts managed

False positive → false alarm, False negative → event missed (very bad)

	Advantages	Disadvantages
Host-based IDS	• Effects of an attack are clearly visible and therefore detectable (e.g. detect file modifications) • No problems with encrypted network data • Few false positives due to context information (OS, apps)	• Limited view on one host • Puts additional load on the monitored system • Cannot be easily hidden and are therefore attackable • Trustworthiness of a sensor is questionable after successful attack → gets useless if you control IDS, sensors can be tampered
Network IDS	• Broad view of entire activity in monitored segment • No additional load for monitored systems • Sensors can be well protected (passive monitoring, no IP address needed)	• Problems with encrypted traffic • Result of an observed attack attempt are hard to see • Require additional hardware components • Difficult to cope with large traffic volumes (Gigabit Ethernet) • Difficult to get all traffic in switched networks ↳ might be routed another way

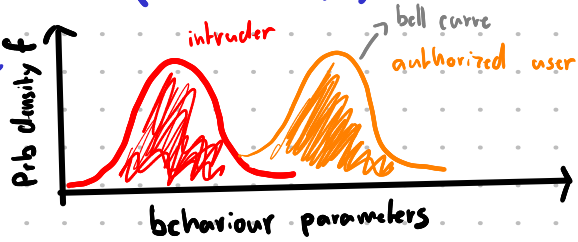
Hybrid IDS: obviously a combination of the two and what is most likely in use

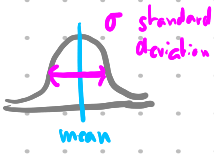
An IDS can be very small (1 device monitored) to huuuge

IDS challenges:
- Minimise false positive/negative
- cope with large amount of data
- where to place sensors
- keep system up to date (daily work)
- playbook for alerts
- Human resources

Anomaly detection is not as easy as it sounds (even with AI)

- reports dont match an expected statistic
  ↳ alert raised if computed statistic don't match expectation
  ↳ Wow does this graph look smart :O
  ↳ done with threshold metrics, statistical methods or markov model

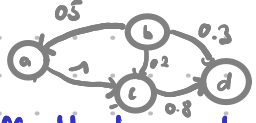




Threshhold: between m and n events → account lockout expected to occur. If outside of that range → bad

Statistical methods: mean and standard deviation calculated and if it is outside of the prediction → harmful
↳ challenges: behaviour can be modeled, follows gaussian distribution, fast enough?

Markov model: · past state effects current transition
· establish valid sequences
· training data needs to cover all possible cases → difficult to get



Protocol anomaly: attacks with by deliberately misusing protocols. Fairly easy to detect as normal users don't violate protocols and they are well defined

The methods above assume some statistical distribution. ML doesn't do this

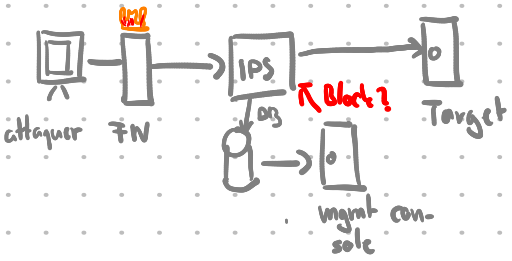
P. 32-37 not covered

IDS and logging architecture is basically the same

IDS Responses

Alert raised, what now?
- Do nothing
- automated response (close port, lock user, TCP Reset) or block IP
- manual intervention → slow

IPS: Intrusion prevention System:
- Traffic flows through the sensor
- does active interception
- must be quick → delay

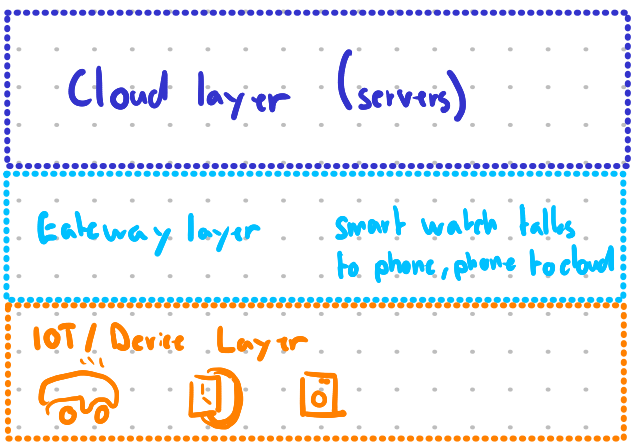


IDS Evasion:
- See PTSec zsm for Vortrag of Daniel
- Options: Fragmentation, change order, encoding
  ↳ Countermeasures: check packets of a session instead of individually
- The broken ZIP

9) IOT Part 1+2

IOT device is a single-purpose device that connects to the internet

Actuator is the thing responsible for controlling a system or mechanism



Gateway Layer needed for: handle sheer numbers, energy costs, TLS

Mirai Botnet attacked a lot of IoT devices. See PFSec for more details.

Learnings: change PW (forced by manufacturer), disable ports, patch IoT (if possible)

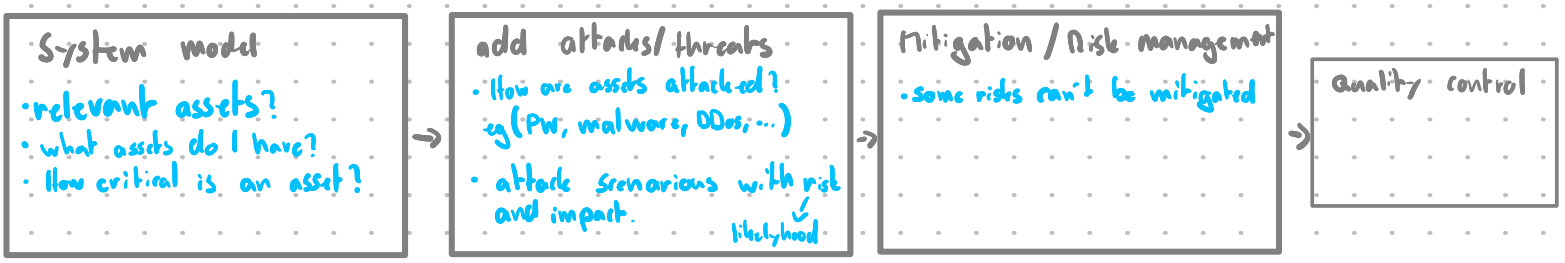
IoT Security economics:

Suttons Law: "I robbed banks because that is where the money is"

Actors have different insentives → important to consider them

- Script kiddie: Fun?
- Hacktivist: for the good
- Cyber criminal: money
- Nation state: influence
- Insider
- Terrorist

Threat Modelling is the process of analysing threats from the pov of an attacker. Different Models exist like STRIDE or OWASP IoT



Design Principals for Security architecture

1) Open Design Principal: how cryptography works is public and doesn't make it less secure (↑ arguably more)
2) The simpler the better
3) Fail-safe default: Default deny principal
4) Separation of priviledge principal
5) Complete Mediation (↑ Schlichtung) Principal: never cache access control decisions. Best: authenticate every thing
6) Least priviledge principle
7) Least common mechanism: system to work as independantly as possible
8) Defense in depth: more layers → more secure, swiss cheese principal
9) Trust no one! → not ideal in Real Life arguably
10) Secure the weakest link: Secure the door first before worrying about someone bulldozing the wall

OWASP IoT Top 10

1) Weak, guessable, default or hardcoded pw
2) Insecure network services: CVEs not patched on IoT device
3) Insecure Ecosystem Interface: Stuff like undocumented, forgotten test API left in code

4) Lack of secure update mechanism : eg. not checking what software is installed

5) Use of insecure or outdated components

6) Insufficient privacy protection

7) Insecure Data Transfer and storage eg: no HTTPS

8) Lack of device management : Excel sheets are used too much

9) Insecure default settings (and why change if it already works)

10) Lack of physical hardening : eg easy way to plug in USB

How to secure the IoT cloud

- Authentication must scale
- End to end integrity
- Traffic should not look like DDos
- Cloud services only to communicate with legitimate devices → TPM chip
- Use encryption

11 IOT Security part 3

Little devices (small resources) : Protect them with gateways and Firewall !

- Turn off necessarily ports, run without admin permission, reboot often, updated securely, authenticate all communication

Bigger devices : treat them like to a laptop

- To protect the usual , advantage that iot devices don't move

Patching : Don't fire and forget , prepare for device management from the start.

Smart IoT devices can actually tell a lot about you:

- Food consumption, Healthiness, vacations and presence, religion, location, ...

Access to PII (personal identifying info) must be:

- Access controlled : login, 2FA
- Anynomized if specific data is not needed

Privacy controls in IOT

- Secure multi party computation where inputs are shared (not what a person chose), instead only the computed part (output) is learned.
- Soft measures like GDPR should protect you aswell

12) Privacy and anonymity

Need for anonimity: internet is a public network, routing is public and the encryption only hides the payload

Privacy is the right of an individual to control his/her personal data.

Data minimization / Hard privacy: provide as little data as possible

Laws, regulations / Soft privacy: laws that make the data controller responsible
↳ GDPR

Some GDPR principals (not complete as mostly skipped): purpose limitation, storage duration limitation, use encryption, accountability, right of access and erasure

Consent must be freely given → cookie banner

GDPR applies to companies doing business in the EU, CH has its own Datenschutzgesetz

Anonymity is the state of not being identifiable in a group.

- anonymous is not possible by yourself
- Hide activities among other similar activities

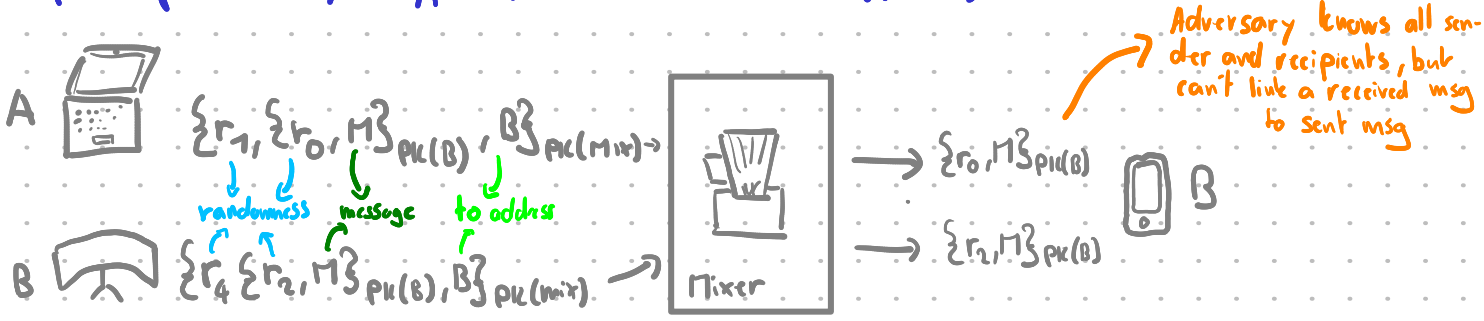
Unobservability - very hard to achieve: action / event is indistinguishable from any other item

How to attack anonymity:
- snoop traffic and analyze who talks to whom
- compromise routers
  ↳ hard to tell which are compromised

Chaum's mixer (1st gen of privacy protocol)

David Chaum in 1981 introduced a untraceable email service. Same principal for other services possible too.

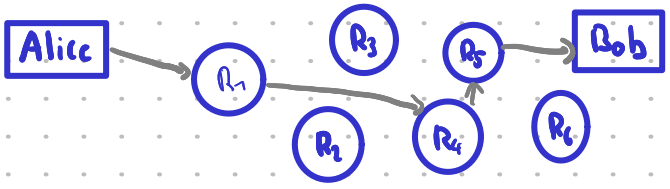
Concept: public key crypto + trusted re-mailer (mixer)



For a return address M includes $\{K_1, A\}_{PK(mix)}$

Usually messages are sent through a sequence of mixers → Mixnet. Mixers are normally spread across juristical boundaries to make it more difficult to compromise all servers. Many users and high traffic improves anonymity.

1st Gen of onion routing



$\{M\}_{PK(Bob)}$

$\{B, \{M\}_{PK(BOB)}\}_{PK(R5)}$

$\{R5, \{B, \{M\}_{PK(BOB)}\}_{PK(R5)}\}_{PK(R4)}$

⋮

Messages now have a high latency. Routers can also resort and work with buffers.

+ High degree of avonymity ; - high latency
- no real time msg
- router might not be online -> msg drop

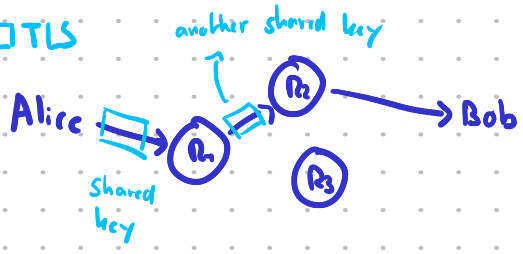
2nd Gen onion routing (Tor)

Small or no re-sorting buffers resulting in low latency

+ allows interactive services like instant msg or ssh

- strength of avonymity drops, especially during low traffic

☐ TLS



$\{M\}_{PK(B)}$

$\{B, \{M\}_{PK(B)}\}_{K_{R2}}$

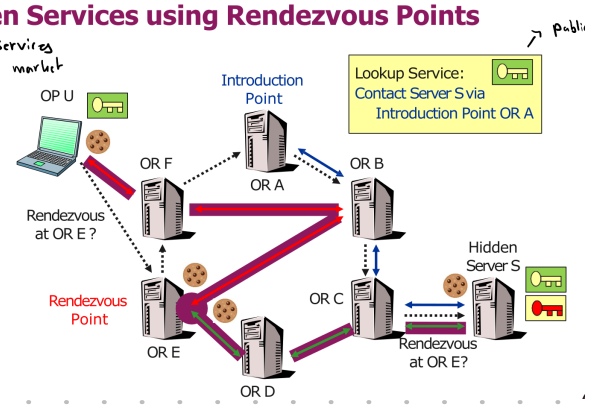
$\{R2, \{B, \{M\}_{PK(B)}\}_{K_{R2}}\}_{K_{R1}}$

↓

R2 is the recipient of the paket received by R1

Hidden Services using Rendezvous Points

- Chat services
- Hidden market



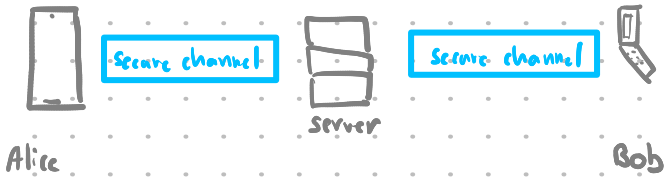
Dining cryptographers : Either NSA pays, or one of them but wants to remain avonymous

1) Flip a coin and show it to the person on the left.
2) Announced if two coins are the same. If he is the payer -> lies
3) odd "same" -> NSA, even "same" -> one of them, but remains avonymous

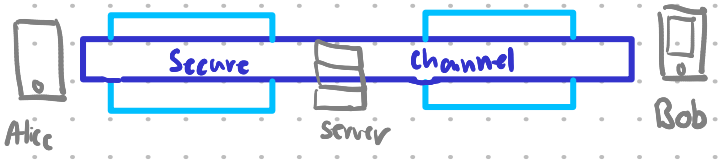
Impossible to tell who payed, even with unlimited computing power.

13) Messenger Security

In-Transit encryption:



End-to-End encryption:



TLS is for active connections, DH needs the clients to be online
↳ what about group chats? What about offline devices

Messaging Layer Security

MLS is built for groups. Group secret is an AES key. Each Epoch generates a new key

Perfect forword Secrecy

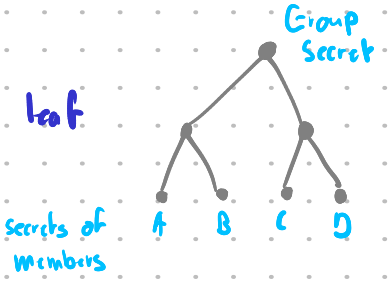
Charlie doesn't know about past convo

Bob doesn't know future convo

Key ratchet: Evolve current key and delete old one (and you can't go back)
↳ Problem: no new randomness so Bob (the fucker) can also compute it

⇒ Double ratchet: KEM (key encapsulation mechanism) brings new randomness
↳ Problem: only for two party setting

⇒ Tree based key structure. Ratchet the key at the leaf



Limitations of E2EE: often unencrypted on end device (if unlocked)
↳ Spy ware can exfiltrate

Backups are often not encrypted