2017-2 Network Security

85540 characters in 13152 words on 2293 lines

Florian Moser

February 20, 2018

1 network summary

1.1 layers (vertical)

layers stacked up on one another (802.11 → IP → TCP → HTTP)
lower layers are outside and wrap the next upper layer protocol
each layer adds its own header
higher layers uses the services of the next lower one (encapsulation)
more evolved in practice, with trailers, encryption, compression

OSI 7 Layers
physical (sends bits as signals)
data link (sends frames of information)
network (sends packets over multiple links)
transport (provides end-to-end delivery)
session (manages task dialogs)
presentation (converts data representations)
application (provides functionality to users)

internet reference model
application (SMTP, HTTP, RTP, DNS)
transport (TCP, UDP)
internet (IP)
link (Ethernet, 3G, Cable, DSL, 802.11)

1.2 protocols (horizontal)

protocols of same kind communicate with each other

ethernet
preamble 8
destination address 6
source address 6
type 2
data 0-1500
pad 0-45
CRC checksum 4

IPv4
version 4, IHL 4, DS 8, total length 16
ID 16, empty 1, DontFrag 1, MoreFrag 1, FragOffset 14
TTL 8, Protocol 8, Header checksum 16
Source Address 16
Destination Address 16
Options (0 or more words d∗32)
Payload

IPv6
8 groups of 4 hex digits
omit leading zeros
shorten all-zero group to single zero
shorten multiple zero groups using double colon (only allowed once)

1.3 component names

application (or app, user) uses the network (skype, browser)
host (or node, source, sink) supports apps (laptop, server)
node (or switch, router, hub) relays messages (AP, cable, modem)
link (or channel) connects nodes (wires, wireless)

1.4 hardware

middleboxes
sit inside network, but perform more than IP processing
can change the outer layer (from ethernet to 802.11)
NAT, firewall, intrusion detection system

NAT
connects internal with external network
many internal hosts using only a few external addresses
motivated by IP address scarcity
map IP & ports of inside hosts to different outside ports/IP combinations
needs to lookup and rewrite IP packet

1.5 routing

decentralized, distributed node run same algorithm
operate concurrently, communicate using messages
node/link/message failures possible

1.5.1 needed properties

correctness (find working path)
efficiency (use bandwidth well)
fair (don't starve any nodes)
fast convergence (recover quickly after changes)
scalability (support large networks)

1.5.2 target bandwidth allocation

efficient (most capacity used, but no congestion)
fair (every sender gets reasonable share)
max-min fairness (bottled flows get equal share)

1.5.3 algorithms

dijkstra
extract the node with the lowest distance
add link N to the shortest path tree
relax the distances of neighbours of N by lowering to the best estimate

distance-vector DV
routing table contains nodes, distances and next hop
nodes adapt table if better connection found by broadcast
more scalable than LS, but slow convergence after failures

link-state LS
nodes flood topology with link state packets
nodes compute own forwarding table
faster than DV, typically used in practice

1.6 ICMP

on routing error, ICMP sent back to source in IP packet

exact
ICMP header with type, code
ICMP data with start of offending packet, the IP header first

traceroute
sends probe packet with increasing TTL starting at 1
logs ICMP errors, and therefore knows the path the packet is taking

ping
ping request/reply (type=8/0)
code is always 0, if reply received then host reachable

destination information
destination unreachable (type=3)
code is 0/1 if net/host is not reachable
code is 4 if fragment is too big (MTU is set too high)

path error
time exceeded (type=11)
code is always 0, if TTL of packet is maxed out
protects against forwarding loops
used by traceroute to find path of packet

1.7 IPsec

operates at network layer
provides data integrity, secrecy
multiple algorithms available to secure connection
encrypts traffic between two "Security Gateway"s

Security Association SA
an end-to-end configuration
contains security identifier

modes
tunnel (IPsec packet inside IP packet)

transport (header with SA number)

IKEv2
established tunnel
protocol which provides authentication
agree on security proposal, keys setup

headers
authentication header (integrity checks, sequence number)
encapsulating security payload (secrecy)

investigates issues
packet type dependent traffic handling (other paths chosen)
port NAT (packets modified in transit)
modified sequence numbers (DoS attack, ISP made mistake)
man in the middle (downgrade attack, response time too slow)

1.8 TCP

secure connection at network layer

exact
source port 16, destination port 16
sequence number 32
acknowledgement number 32
tcp header length 4, various 12, window size 16
checksum 16, urgent pointer 16
options (d*32 words)

connection establishment
sender and receiver need to be ready to receive, agree on parameters
with three way handshake
client sends SYN(x)
server replies with SYN(y)ACK(x+1)
client replies with ACK(y+1)

connection release
active FIN(SEQ=x), passive (ACK=x+1, SEQ=y)
passive FIN(SEQ=y, ACK=x+1), active (SEQ=x+1, ACK=y+1)
each FIN/ACK closes one direction

sliding window
building on stop and wait
sends while LastFrameSent - LastAckRecieved < Window

flow control
slow down enthusiastic sender by sending available buffer space
sends WIN together with ACK of available space
may resends same ACK if buffer increased

retransmissions
if timeout occurs, resend packet
adapt timeout based on last data

TCP slow start
increasing cwnd with each packet received (doubling each timeslice)
use till congestion window reached, then continue with AI

Additive Increase, Multiplicative Decrease (AIMD)
AI to reach max transmission speed (increase slowly)
MD to avoid congestion detected (if detected, half bandwidth)
leads to characteristic Sawtooth pattern
converges to allocation which is fair and efficient

detect congestion
packet loss (easy but too late, used by TCP NewReno & Cubic TCP)
packet delay (know early but complicated, used by Compound TCP)
ERN (know early but router needs to explicitly set notification)

2 Crypto Refresher

2.1 basic definitions

anonymity
keep identity (name / identifier) of a protocol participant secret

privacy
the control over how information about oneself is shared

confidentiality
extension of privacy
defines how identifiable data is treated

secrecy
more general than confidential, privacy, anonymity
keep data hidden from unintended receivers

integrity
property of local/stored data
unchanged and syntactically correct
prevents unauthorized, improper changes

non-repudiation
sender cannot deny sending of message
initial receiver can pass message with proof to third party

authentication
of property ensures identity of sender verified and data integrity
of entity verifies identify of another protocol participant
of data ensures data originates from claimed sender

2.2 cryptographic primitives

asymmetric (public-private key)
public key to verify/encrypt
private key to sign/decrypt
public/private key systems such as RSA
digital signatures
more powerful than symmetric, but much slower

symmetric (shared-key, same-key)
secret key to encrypt/decrypt
block ciphers (pseudo-random permutation PRP)
stream ciphers (pseudo-random generators PRG)
message authentication codes (MAC)
use asymmetric to setup secure key

hash
provides integrity
no keys needed

MAC
provides integrity, authentication
uses symmetric keys

digital signatures
provides integrity, authentication, non-repudiation
uses asymmetric keys

2.3 symmetric primitives

2.3.1 one-time pad

use random keystream for each message (XOR with plaintext)
perfectly secure if bytestream never reused

2.3.2 stream cipher

to archive secrecy
very fast (faster than block ciphers)

using PRG (Pseudo Random Generator)
Initialization Vector IV, secret key k
cipher = plain XOR PRG(k, IV)

keystream reuse attack
if same keystream used for two different messages
XOR two cipher text gives XOR of two plain texts

ciphertext modification attack
flip a bit in cipher will flip a bit in plain
integrity of message needed

general security question
is the bitstream reused?
can I change a bit?

2.3.3 block ciphers

to archive secrecy
take fixed size of plaintext
encrypt each block separately

DES
computation power of bitcoin larger than needed to break DES

AES/Rijndael
works well and fast
blocksize 128
key size 128, 192, 256
Intel and AMD implement special instructions for AES
split message M into m_1, m_2, ..., m_n
encrypt each block mode specific

semantic security
advisory can guess value of single plaintext bit at most random
even if same message is encrypted, the cryptotext looks random each time

cannot infer if newly encrypted message is same as before

2.3.4 block cipher modes

Electronic Code Book (ECB)
block encrypted with key K
simple to compute
but same plain equals same cipher, restructure possible

Counter Mode (CTR)
IV encrypted, XOR with block, new IV = old IV + 1
semantic security
but altered cyphertext only influences one block

Cipher Block Chaining (CBC)
block XOR IV, block encrypted (is new IV)
semantic security
but altered cyphertext only influences two blocks
if bits flipped in C2, then P3 has same bits flipped

Cypher Feedback (CFB)
IV encrypted, XOR with block (is new IV)
semantic security, altered cyphertext influences all following blocks

Output Feedback (OFB)
IV encrypted (is new IV), XOR with block
semantic security
but altered cyphertext only influences one block

Galois Counter Mode (GCM)
gives both secrecy and authenticity

2.4 message authentication codes (MAC)

to archive authenticity, needs a symmetric key

hash-based MAC (HMAC)
choose hash function, adapt key K to hash block size
secure even if strong collision resistance broken
HMAC[K, m] = H[K XOR 0x5c5c... || H[K XOR 0x3636... || m]]
(pads allow for more entropy, maximize hamming distance)

block-cypher based MAC (CMAC)
encrypt message with block cypher
use last block as MAC

multiple hash functions
can AES encrypt file, use as many bits as needed from cypher
secure encryption implies same properties as good hash function

2.5 asymmetric primitives

diffie-hellman (DH)
public values are large prime p, generator g
private values are a, b
A to B → r_1 = g^a (mod p), B computes key = r_1^b
B to A → r_2 = g^b (mod p), A computes key = r_2^a
Eve cannot compute g^ab
man in the middle possible (advisory can sit between A and B)
works because of discrete logarithm problem

RSA algorithm
choose p,q as large secret primes
pick public e, compute secret d such that e∗d mod ((p-1)(q-1)) = 1
publish public key N=p∗q, e (and keep p,q,d secret)
create signature with v = M^d mod N
verify signature by M = v^e mod N

Encrypted Key Exchange (EKE)
A,B share private p, hash function H, use it to calculate K = H(p)
A,B have private values a,b respectively
use EKE to calculate session key K' = H(g^ab)
verify with nonce that other party was successful
A to B → g^a ENC K
B to A → g^b ENC K, nonce_b ENC K'
A to B → (nonce_a, nonce_b) ENC K'
B to A → nonce_a ENC K'

2.6 cryptographic hash functions

maps arbitrary length input to finite length output

2.6.1 properties

one-way
give y = H(x), cannot find x' such that y = H(x') (no inversion)
find x' after 2^(n-1) on average, probability for each try is 1/(2^n)

weak collision resistance
given x, cannot find x' != x such that H(x) = H(x')
same complexity as one-way

strong collision resistance
cannot find x != x' such that H(x) = H(x')
root(2^(n/2)) because can test against any hash (birthday paradox)

2.7 hash chains

robust to missing values, infeasible for attackers, efficient to verify
pick secret and public one-way function (hash H)
hash n times (r_i = H(r_(i+1))) and publish r_0
used as cheap one-time passwords
used in router protocol (can't claim shorter distance)

enhance
security lowered with each more revealed hash chain entry
attacker can try to create a collisions
loose logn bits of entropy with each new revealed entry
simply add index to hash like r_i = H(r_i+1, i+1)

2.8 merkle hash trees

authenticate sequence of data values D_i
build up tree by hashing two together T_3 = H(D_0 || D_1)
walk up the tree by continuously hashing nodes, sign root
to authenticate D_i, accompany it with nodes to reach root
if signed root node can be reached, then D_i authenticated

2.9 power of two conversions

remember 2^1 till 2^10

calculation
2^n = 10^m
n = (m/3) ∗10
m = (n/10) ∗ 3

basic big numbers
seconds per day is 2^16
seconds per year is 2^16 ∗ 2^9 = 2^25

3 Introduction to blockchains

3.1 bitcoin

first block mined in 2009, first commercial pizza bought for 10'000 bitcoins
used for criminal activities, because hardly traceable
original whitepaper by Satoshi Nakamoto describes only main operation
reference implementation is de-facto specification
forks expected to appear as reference implementation holds no authority

traditional digital cash
first paper (1982) by david chaum introduced anonymous coins
bank only verifies that same coin is not used two times
commercialized as DigiCash but not a huge success
improvements included offline transaction, and breaking into smaller coins
all require trusted authority, the bank, bitcoin removes central authority

core concepts
proof of work to combat spam / DDoS attack
public ledger to prevent double spending (transactions are state)

innovation
archives consensus without trusted authorities or preassumed identities
under such constraints the general problem of consensus is impossible
uses additional assumptions about incentives & networking
lessons learned from bitcoin may help to create better tools

theoretical problems
difficult to model in precision (relies heavily on socio-economic factors)
hard to argue about soundness

3.2 main components

transactions
input contains a scriptSig with public key and signature
inputs are combined, each input group gives two outputs
one output for the real receiver, one output for rest back to the sender
output contains a scriptPublicKey which signs a script
script must evaluate to true for the output to be valid
scripting is simple stack-based language (not turing complete)
valid transactions are included in block
block is put in public ledger (block chain)

transaction is "confirmed" when block deep inside chain

ownership
each user has wallets containing a public/private keypair
knowledge of private key gives ability to spend or redeem outputs
user state represented as set of transactions using the scripting language

nakamoto concensus
only signing does not prevent using same output multiple times
therefore all transactions are saved into public log (ledger), chain
can extend the chain by solving computational puzzle (mining)
reject wrong solutions, and append to longest chain
multiple chains may exist (forks) if multiple solutions
eventually one of those chains will be longer (eventual consistency)

mining
find nonce for H(H(previous_block) || transactions || nonce) < target
target is adjustable (the lower the harder), solving by brute force
miners finding a valid nonce get the block reward & transaction fees
block rewards will half every four years till 2140, then cease to exist
transaction fees will stay, amount to several bitcoins
proof of work (puzzle) with an incentive to work (fees)

peer to peer network
used to broadcast new transactions and blocks
protocol works by flooding
INV message to advertise block
GETDATA message to request full block data
network has low latency and can't block any participants
any node connects to eight other nodes as outgoing connections
any node accepts about 100 incoming connections
10min delay, 10'000 nodes in bitcoin network

3.3 bitcoin analysis

3.3.1 needed for stability

eventual consensus
all compliant nodes eventually agree on longest chains

exponential convergences
probability of forks is exponentially proportional to length of chain

liveness
valid transactions will be added to the chain

correctness
in the longest chain all transactions are valid

fairness
fraction f of computing power should be able to mine 1/f blocks

3.3.2 properties

stability
if (majority honest) then consensus, convergence and liveness
but needs good network (latency lot lower than block discovery)

privacy
users create many pseudonyms, this guarantees strong privacy
merchant can create new pseudo for each received payment
payer may has to combine multiple sources to pay

performance
7 transactions per seconds (tps)
6.5s delivery time of new block (95% reception at 40s)
can adjust parameters to increase to 60tps
slower generation rate + smaller block = higher security

upgrade
using soft forks (backwards compatible, only few changes possible)
using hard forks (enabling new kinds of transactions)
har fork bugfix from 2013 won only after 24 blocks

3.3.3 attacks

large mining power (more than half)
ignore blocks found by others because own chain will grow faster
refuse to include certain transactions
introduce arbitrary forks

block withholding attack (selfish mining)
waste resources of others
miner finds two blocks, waits with publish till other have found one
higher expected payoff

eclipse attack
isolate (eclipse) node from the rest of network
monopolizing its incoming and outgoing connections
enable targeted double spending, easy with access to cable

off path attack
get the victim to only connect to attacker nodes
advisory does not need to be directly connected to victim
influence victims view of network using other nodes to send fake data
on victim restart, will likely connect to advisory nodes only

message delaying
inform victim late about changes in network
delay delivery of block, instead of relaying them
only requires single connection, exploits scalability measures

routing level attacks
delay/disconnect victim from network
possible because BGB hijacking
completely disconnect / delay messages for up to 20 minutes

deanonymise participants
mapping money flow to individuals is often possible
parse through blockchain, link transactions to identity cluster
then try to make trade with one of those clusters to find out identity
broadcasting to network leaks IP (can use TOR to mitigate)
then link together multiple pseudonyms used in one session

3.4 blockchains

permissionless
anybody can mine
can scale to a big number of nodes (but low throughput)
low performance compared to more centralized systems (like VISA)

permissioned
blockchain consensus is maintained by predefined entities
can only scale up to small number of nodes (but good throughput)

3.5 examples

bitcoin
permissionless & special-purpose
only for money

ethereum
permissionless & general (turing complete byte-code language)
currency called ether can be transferred to users/contracts
programs are called "smart contracts", fee paid on execution
can invoke functions of other contracts
transactions contains state of financial application

hyperledger
permissioned & general
supports smart contracts
can select small number of validators to validate next block
dramatically increased performance

other applications
replace database with permissioned blockchain
replace property registry; each entity can run its own validators in it

3.6 research

blockchain address certain problems but not perfect
blockchain does not ensure data secrecy
blockchain hype to extreme

open questions
security (attacks on mining, network, incentives)
consensus (is blockchain really the way to go?)
partially-centralized (control chains)
trusted computing (build stronger chain)

4 DNS

4.1 concept

globally distributed database (loosely coherent, scalable, dynamic)
no single computer has all DNS data
maps names to resources of various types (URL to IP's for example)

4.1.1 attackers perspective

helps to setup services which are hard to shutdown (botnets)
helps building hidden channels (can send UDP through DNS)
distributed storage system (can be used to distributed files)
abused for DDoS attacks

abused for impersonation attacks

4.1.2 consists of

namespaces (hierarchy)
servers (making namespaces available)
resolvers (query server)

4.1.3 hierarchy

from right to left, each dot separates a domain
. ("dot") is root level (root servers, controlled by US)
TLD top-level domains controlled by registrars (generic/country TLD)
SLD second-level domains controlled by private entities
FQDN fully-qualified domain (mail.ethz.ch)

4.1.4 resolution

ask ISP caching resolver, which then asks from root to SLD
caching only valid as specified by TTL
dig 8.8.8.8 url.ch

4.1.5 protocol

simple UDP/TCP protocol
no encryption, authentication nor integrity built into
DNSSec is a secure extension, but not adapted largely
query section (with target url)
reply section (with target IP
authorative section (with nameserver URLs)
additional section (more IP/URL mappings, including for NS)

4.1.6 name server

authorative for a specific zone (ETH nameserver for ETH domains)
cache response for queries as long as TTL is valid

4.1.7 resolvers

stub resolver (library which contacts recursive resolver)
recursive resolver (iterative DNS resolution)

4.1.8 resolution

request has 16bit transaction id TXID (generated on device)
replies can be matched using the transaction id
replies contain TTL
responds with NXDOMAIN if not found
responds with most specific authority found

4.1.9 record types

address record (A, AAAA) to map FQDN to IP
pointer record (PTR) to map IP to FQDN
name server record (NS) advertises name servers for zone
start of authority (SOA) advertises attributed of zone
canonical name (CNAME) may defines alias for FQDN
mail exchanger record (MX) to define mail server URLs
text record (TXT) for untyped storage

4.1.10 fundamental DNS problems

session id does not have enough entropy, or is predictable
amplification due to large response for small request
open resolvers respond to any query even from outside network
content in additional section of DNS protocol is cached

4.1.11 name server roles

authorative name servers
non-recursive response with data it is authorative about
responds to queries from everywhere

cache/recursive resolver
attempts to resolve fully
only responds to queries from network

4.1.12 attacks

phantom domain attack
advertise domain on attacker NS, which on purpose responds slowly
ask real DNS server for resolving
real DNS server hangs waiting for response from attacker NS

URL access check
ask DNS server, if TTL has low value URL has been accessed before
can abuse to see if URL has been visited without owning the domain

distributed reflection
create TCP packet with spoofed victim IP
DNS sends big response for small request to victim

DNS spoofing
reply faster than DNS server (DHCP forge, random broadcast)
set high TTL to forged data
adds more URL/IP in additional section of DNS packet
poison shared DNS cache with forged DNS entries
modify hosts file to point to attacker NS
manipulate DNS settings to point to attackers NS

DNS/domain hijacking
hack the domain registrar / abuse weak passwords
change domain NS and set high TTL

DNS tunneling
bypass firewall by sending traffic over DNS
mostly fixed nowadays

4.1.13 attack mitigations

distributed reflection
reject packets with IP unreachable from actual path
establish response rate limit per IP
authoritative nameservers disable recursion, only respond for own zone
recursive resolvers limit to authorized clients from own network

cache poisoning
detect poisoning with a monitoring service
set high TTL for own records (attack retry time longer)
protect with SSL certificates (need to have capable users)

4.2 botnet control with DNS

botnet must have reliable way to be controlled
fixed IP allows to identify bot agent & controller and is easy to block

4.2.1 botnet architecture

main controller / bot master, shielded by layer of intermediaries
layer of redundant bots to hide bot master
network of managed bots which perform tasks

4.2.2 control architectures

use single domain
set low TTL, multiple IP addresses (A records)
hart to shutdown because of DNS caches, but still weak

use IP flux
frequent change of IP related to FQDN
TTL to low value, provide array of valid IP addresses

use domain flux
frequent change of FQDN
domain generation algorithm DGA to computes list of domain names
attacker registers one of the possible domains
bot attempts to connect to any of the generated domains
defender cannot simply buy all of them

4.3 DNSSec

attempts to add security, while being backwards compatible

4.3.1 provides

origin authentication of DNS data
authenticated denial of existence (so can't add stuff to the message)
integrity (but not availability or confidentiality)

4.3.2 how

all DNSSEC zone data is signed with private key of zone
response consists of signed record set
can be verified with trusted public key

4.3.3 problem with DNSSec

DNS resolves to found, not_found, but cannot express forged
deployed at ISP level
need to manage master keys

4.3.4 new fields

resource record signature RRSIG
contains DNSSEC signature for record set
verified using the DNSKEY

public key record DNSKEY
public key used to verify DNSSEC

delegation signer record DS
references a DNSKEY (glues the chain)
lives in the parent zone, singed by parent or other trusted key

next secure record NSEC
contains link to next record name in zone (non-existence proofs)

so if A,C exist and asked for B, return A NSEC B

5 SSL/TLS Public-Key Infrastructure (PKI)

5.1 introduction

security in network stack
PGP in application layer
SSH, TLS in transport layer
IPsec in internet layer
AES, WPA2 in link layer

security requirements
secrecy to prevent eavesdroppers to learn sensitive information
entity and message authentication to prevent alteration / injection

browser warnings
41% for facebook, youtube, google
61% clock, 13% antivirus, 13% captive portal, 3% unknown

5.2 PKI

public key infrastructure provides a way to validate public keys

trust establishment
trust root which cryptographically transfers trust
trust depends on size of trust root (how reliable)
trust depends on number of malicious entries tolerable

certificate authorities (CA)
trust roots in TLS PKI which can sign keys of lower entities

possible trust roots
monopoly model with single trust root (DNSSEC, BGPSEC)
oligarchy model with numerous roots of trust (SSL, TLS)
both implementations lack update of trust roots
who controls root vs weakest-link security

X.509
from 1988, standard format
defines structure of PK certificates with data & signature section
contains valid from, valid to, and permissions (any, only sign, ...)

5.3 TLS PKI

levels of trust
no ssl/tls
domain validation DV (common case, verify that you own domain)
organization validation OV (more info about owner)
extended validation EV (a lot of checks till issued)

multi-domain certificates by CDN
CDN need https but sit between user and webpage
CDNs therefore obtain certificate for multiple domains (from CA)

problems
CA's can sign any URL/email, may abuse privilege
lots of CA's means large root trust base but weakest link security

5.4 improve TLS

challenges
cannot tolerate additional latency during ssl handshake
certificate has to be immediately usable and verifiable
need to be able to revocate certificates
users shouldn't decide if certificate is legitimate

CAP theorem
Consistency, Availability, Tolerance to Partition
need to sacrifice one

short-lived certificates
only very short livetime to avoid revocation issues

Lets Encrypt CA
provide free TLS certificates, sponsors like mozilla
automate domain validation, issuance and renewal (90 days valid)
uses ACME (automated certificate management environment) protocol

HTTP strict transport security (TSTS)
allow servers to declare HTTPS only in its header
(Strict-Transport-Security: max-age=62631263)
prevents downgrade, SSL stripping, session hijacking

certificate revocation list (CRL)
revoke certificates (because private key disclosed, employee leaves)
url in CA certificates resolves to revoked certificates

online certificate status protocol (OCSP)
addresses issues with CRL
(slow) protocol which clients can use to verify certificate status
clients directly ask at CA, which responds with signed response
browsers treat errors non fatal, rather relay on software update

OCSP stapling
replaces revocation, saves OCSP status inside certificate
server needs to periodically renew its certificate status at CA

DNS-based authentication of names entities (DANE)
goal is to authenticate TLS servers without CA
add TLSA entry to DNS containing constrains
add cert constrains (directly specify certificate)
add CA constraints ("only trust those CA's")
add trust root assertion (specify new trust root)
relies heavily on DNSSEC (which has single trust root)

HTTP public key pinning (HPKP)
server sends set of public keys to client for future connections
in HTTPS header Public-Key_pins: max-age=9123981, pin-sha256="..."

5.5 certificate transparency (CT)

hold CA publicly accountable for all issued certificates
do so without introducing yet another third party

Integrity Log Server (ILS)
holds log of published certificates publicly available

technical
CT log is append-only list of certificates
ILS can append any new entries, then signs root node
data structure used is merkle hash tree with certificates as leaves
does not endorse certificates, only makes them publicly available
verify if entity is part of the tree is fast (hash till at root)
verify if tree is append only is efficient

SCT (Signed Certificate Timestamp)
when CA adds certificate to ILS it generates a SCT
include SCT in the certificate directly to avoid ILS roundtrip
as the SCT generation may takes some time there are different approaches

X.509v3
CA asks ILS for SCT with pre certificate
adds SCT to certificate before issuing it

TLS extension
domain submits SSL certificate to ILS
adds SCT to handshake

OCSP stapling
CA submits SSL certificate to ILS
domain queries CA for SCT, then adds it to handshake

participants of CT
monitors of CA watch logs for suspicious activity (also read-only backups)
certificate owners verify that no others have certificates on same name
auditors of browsers verify overall logs integrity, redo log proofs
auditors & monitors communicate to detect forked logs

security of CT
browsers demand SCT for certificate, may contact ILS to verify
deployable, no change to web server required
MitM and bogus certificates can still happen, but will be public record

5.6 Attack-Resilient PKI (ARPKI)

reduce trust in single component as domain creates own security policy
properly define events as key change, revocation, switch of CA
address adversarial events (CA, logs are compromised, attacked)

requirements
checks and balances (distributed trust, parties monitor each other)
brief error periods (such as compromise, unavailability)
trust agility (domains & users can pick entities they trust)
privacy (connection only revealed to client & server)
efficiency (no setup time increase)
usability (secure by default)

assumptions
browsers store authentic public keys of CA, ILS's
all parties are loosely time synchronized
advisories main goal is MitM

entities
client (browser) establishes connection with domain

ILS logs domain certificates in Integrity Tree IT (public)
CA signs domains public keys and monitors ILS behaviour

communication
domain obtains certificate with multiple signatures from multiple CA
CA act as validators to survey operations by ILS
n parties needed (1 ILS, n-1 CA) for valid certificate

communication flow for n=3
domain writes CA
CA creates certificate, and adds it do a quorum of ILS servers
CA sends to CA_2
CA_2 confirms certificate and quorum of ILs
CA_2 sends certificate to CA
CA sends certificate to domain

6 TLS

6.1 SSL TLS

history
SSL Secure Sockets Layer by netscape, full of bugs
TLS Transport Layer Security, standardized
TLS 1.2 deployed currently
TLS 1.3 finalized in 2017

fundamental security requirements
secrecy, prevent eavesdropping (passive attack)
authenticity, prevent alteration of entity & message (active attack)

security assumptions
CA assumptions (no key leaks, no singing abuse)
crypto assumptions (no hash collisions, no broken algorithms)
browser assumptions (certificates up to date / not altered, browser secure)
user assumptions (user checks for https, events treated appropriately)

6.2 cypher suite

6.2.1 key exchange metrics

m1 secure to passive attackers (eavesdropper)
m2 secure to active MitM attackers
m3 perfect forward secrecy PFS (prevent backward decryption)
m4 contributory key agreement (both parties add to the key)
PFS needs long-term key to not compromise session keys

6.2.2 cipher spec

cypher algorithm for encryption (RC4, DES40, IDEA, AES, Camellia)
mac algorithm for authentication (MD5, SHA-1, SHA-256)
cypher type (stream, block)
is exportable (true, false)
hash size (0 or 16 MD5, 20 SHA-1, 32 SHA-256)
thesprawl.org/research/tls-and-ssl-cipher-suites

6.2.3 key exchange

to establish session key between client & server
server has public/private keypair K
client/server establishe session key K'

RSA (case 1, server RSA key can encrypt)
client sends session-key to server
server_certificate {URL, K}_KCA-1
client_key_exchange {K'}_K
(m1 m2) fulfilled, (m3 m4) not

RSA (case 2, server RSA key cannot encrypt)
temporary keypair K_2 used to establish session key
server_certificate {URL, K}_KCA-1
server_key_exchange {URL, K_2}_K-1
client_key_exchange {K'}_K_2
(m1 m2 m3) fulfilled, (m4) not

Anonymous DiffieHellman
DH without authentication
server_key_exchange p, g, g^S
client_key_exchange g^c mod p
(m1 m3 m4) fulfilled, (m2) not

Fixed DiffieHellman
DH key of server in certificate
server_certificate {URL, g^s mod p}_KCA-1
client_key_exchange g^c mod p
(m1 m2 m4) fulfilled, (m3) not

Ephemeral DiffieHellman
DH key singed with K
server_certificate {URL, K}_KCA-1
server_key_exchange p, g, g^s, {g^s}_K-1
client_key_exchange g^c mod p
(m1 m2 m3 m4) fulfilled

6.3 exchange

multiple optional requests, marked with (o)
depending on scheme different messages are used

phase 1
server / client agree to specific exchange
C → S client_hello with timestamp, random, cyphers, session id
S → C server_hello with chosen cypher

phase 2
server authentication & key exchange
S → C server_certificate (o)
S → C server_key_exchange (o)
S → C certificate_request (o) (type, list of acceptable CA)
S → C server_hello done

phase 3
client authentication & key exchange
C → S client_certificate (o) (if server requests client certificate)
C → S client_key_exchange
C → S certificate_verify (o) (signing certificate of client, MD5 HMAC)

phase 4
finish up (now client/server share master secret)
C → S change_cypher_spec
C → S finished (send SHA-1 HMAC of exchanged data)
S → C change_cypher_spec
S → C finished

6.4 TLS 1.3

single-roundtrip for initial connection, zero-roundtrip for repeated
after server-hello all messages are encrypted
new key derivation with HMAC
removes optional messages to make protocol easier
new signature algorithms
removed compression because attacker gets information
cleaned-up possible cypher suites

6.4.1 example

C → S client_hello, g^c
S → C server_hello, g^s, certificate, signature, finished, application data
C → S finished, application data

6.4.2 comments

clients assumes server capabilities, try again if guessed wrong
before client finish server already sends non-critical application data
only ephemeral DH allowed
TCP and TLS connection establishment part of QUICK

6.4.3 second connection

client has cached server parameters
C → S client_hello, g^c, server configuration, application data
S → C server_hello, g^s, certificate, signature, finished, application data
C → S finished

6.4.4 attacks

impersonate client
server cannot verify client as no certificates
impersonation of client easily

random number generator RNG
undetectable attack if able to predict state of RNG
decrypt anything as able to predict any generated "secret" keys
client has to send 28bytes of random, but 16bytes would be enough
maybe done to predict state of random number generator
can't trust clients to pick random solely (often outdated)
can't trust servers to pick random solely (could be altered)

dumbing down
replacing cypher spec of client to insecure ones
anonymous diffie hellman perfect for active attacker

7 Denial of Service Attacks and Availability

7.1 DoS

Denial of service (DoS) does resource starvation
deprives of CPU, network/connectivity, memory, diskspace
DDoS is for distributed, large numbers (botnets) attack

7.1.1 general techniques

spoofing (hide/fake the origin)
amplification (in volume and number)
reflection (combine source spoofing and amplification)

7.1.2 attacks types

volume
consume bandwidth within the target network
consume bandwidth between target network and internet
measured in bits/sec
(malformed) ICMP, UDP, IP
reflection / amplification (like DNS, NTP)

protocol
exhaust resources needed to handle protocol on specific devices
saturate protocol state of devices (such as TCP state table)
measured in packets/sec
RST attack (set reset bit to 1, instantly killing connection)
SYN/ACK floods (open connection a lot of connections on server)
fragmentation attacks (invalid fragmentations specified)

application layer (layer7)
attack service or protocol at layer 7
attacks difficult to detect with flow-based monitoring tools
measured in requests/sec
SMTP, DNS, SNMP, FTP, SIP
database connection pool exhaustion (execute long search queries)
slowloris, slow post/read

7.2 compression bomb

attacks memory / storage
file that unpacks to enormous size
wasting / maxing out memory, storage, CPU

compression
transforming data to fewer bits by removing redundancy
lossless exploit redundancies (save repetitions of symbols)
lossy drops nonessential details from source

types
html bombs which define very complex browser structure
image bomb such as compressed GIF or PNG
zip bombs which decompresses to a large file

effects
out of memory during decompression
process reading compressed data takes forever

usages
on any program that inspects traffic / decompresses information
email attachments (anti-malware/gateways/proxies will try to inspect)
webpages (browsers need to allocate memory for images, tables)

mitigation
limit resources for decompression (CPU load, memory, storage size)
timeout (if decompression takes too long)
streaming scan (decompress data on the fly in chunks)

7.3 session state exhaustion

server needs to remember B to act appropriately on ACK
try to exhaust session state table of server

SYN Flood attack
just send SYN packet, but no ACKs
server session overflows
either hangs, removes all active sessions, or denies new ones

mitigation
use function to derive A from B (don't remember A)
store whole state inside session key
but need to prevent session hijacking

7.4 IP spoofing mitigations

ingress filtering
gateway device drops packets with invalid source IP
source-based, needs global deployment

iTrace
every 20'000s packet router sends routing information packet
DDoS victim can reconstruct source of IP attack
but extra packets waste bandwidth, not deployed

IP traceback
routers mark IP packet flags which enable reconstruction of real path
no extra overhead
but requires > 1000 packets, not deployed

7.5 slowloris attack

send GET request, connect to web server, and send request very slowly
server awaits full request before responding
keep many connections open to fill up connection pool
send partial, or very slow requests

mitigation
increase maximum number of clients
limit number of connection for single IP address
impose restriction to minimum transfer speed
restrict length of single connection
protocol sanitation (done by firewalls, proxies, ...)

7.6 mail bounce

mail server generates non-delivery report for each failed recipient
bounce message contains all data from message (including attachment)
set "from" to victim, set "to", "bcc" to non-existing address
amplification by number of error messages times message size

mitigation
at most one error messages for one incoming message
ensure error message is smaller than incoming message
block invalid request as close to the source as possible

7.7 smurf attack

construct ICMP ping packet with victim IP address
send message to IP broadcast range
amplification by number of responding hosts

mitigation
do not forward ping directed to broadcast addresses (router)
do not respond to ICMP broadcast address (host)

7.8 internet of things

small devices produced at large scale as cheap as possible

problems
scarce resources (limited entropy, security, encryption)
standard passwords, credentials in firmware (same for all devices)
unlimited access to internet, longevity, no patches

possible attacks
dictionary attacks to common password like RockYou
extract strings from firmware updates
wiki.skullsecurity.org/Passwords

mitigation
no default passwords, no hardcoded credentials
monitor IoT traffic
support updating and develop patches for vulnerabilities
but medical devices lose the certification with updates

7.9 availability

redundancy
no single point of failure, always n+2 systems running
geographic & internet connection diversity
use microservices communicating on bus (easy upgrade / scale)

monitoring & raid detection
detect issues rapid & automatic failovers
evict failed nodes immediately

failure resiliency
should be able to tolerate various component failures
graceful degradation

long term monitoring
consider that average load may not be sufficient

continuously gather data and look at time series
plan for extreme peaks (over provisioning)

fast recovery
immediate failure detection
alert personnel (use various channels in case one is broken)
recovery plans & staff ready

8 SCION

8.1 internet weaknesses

8.1.1 Spoofing & DDos

impractical to defend with limited money
can easily spoof sender address, abuse internet services
receiver cannot choose how to receive packets

8.1.2 Prefix Hijacking

announce more specific IP so packet is received (easy)
craft BGP packets to create back path (difficult)

8.1.3 kill switches

can halt communication with DDoS, BGP hijacking, DNS redirection
BGPSEC / DNSSEC / TLC certificate revocation

8.1.4 PKI vulnerabilities

single points of failure (like root certificate ch)
security of the weakest link (all CA can issue for all domains)

8.1.5 IP issues

lookup very expensive, worse with IPv6
no transparency for route, limited path availability

8.1.6 other issues

single trust root for DNSsec
unauthenticated ICMP
no clean global framework for PKI
no path verifiability
no protection against DDoS

8.1.7 BGP / BGPsec limitations

availability
frequent periods of unavailability when paths change
slow convergence (hours)
local misconfigurations cause global outages
packet may not be able to forward because routing changed

transparency
poor path predictability and reproducibility

control
no path choice by end points
only single path possible

trust
few trust roots
circular dependency with BGPsec
cannot express policies based on source of traffic

resources
need O(n) messages, need to inform whole internet of change
need TTL to avoid packet looping

BGPsec
even slower, cannot aggregate prefixes anymore
single trust root, cyclic dependency to resolve trust root

8.2 architecture design goals

high availability
if advisory controls parts of network, can't influence other parts
if path consisting of correct routes exists, it is found and used

transparency
chosen path is indeed taken by packet (data plane)
AS can control how they are reached (control plane)
enable multipath routing for availability
ensure response received by real receiver (prevent DOS)
need heterogeneous trust in global environment
need to respect AS forward policies, traffic engineering
balance control among ISP's, senders, receivers

scalability, efficiency, flexibility
BGP convergence / IP forwarding is slow
prevent routers to persist state, stop IP lookup
use simpler routers which only need to decrypt AES
end-to-end principle (end host constructs path & deals with errors)

deployable
need incentives to deploy (competitive advantage)
re-utilize large parts of ISP, simple installation

8.3 SCION advantages

resolves BGP protocol convergence issues
separation of control & data planes (no IP packets)
isolation of untrusted data planes (using ISD's)
path selection architecture (sender & receiver in control)
packet-carried forwarding state (simpler routers)
allows heterogeneous trust root selection

8.4 vocabulary

AS
autonomous systems, part of single or many ISD

Trust Root Configuration TRC
determines trust roots

Isolation Domain ISD
logical grouping of ASes
administered by the ISD Core (selection of ASes called core AS)
governed by a TRC all members accept

packet-carried forwarding state (PCFS)
border routers do not need to maintain forwarding tables

source routing architecture
host has whole network topology and selects arbitrary path

path selection architecture
host can choose from advertised paths, can freely combine them
destination controls which paths are announced
sender controls which announced paths are actually used

8.5 control plane

isolation architecture (restricted route discovery)
determines paths in network
needs ISD's, beaconing, path servers

path types
up path segments (from AS to core AS)
down path segments (from core AS to AS)
core-paths (from core AS to core AS)

PCB (Path Construction Beacons)
core AS floods network with PCB to learn topology
used for both intra-ISD (within ISD) or inter-ISD (amongst core AS)
AS adds name, own Hop fields (In, Out nodes, MACed)
AS adds Peerings to external ASes, sets Expiration, signs all

Intra-ISD Path Exploration (beaconing)
core AS floods the ISD with PCBs, collects routes
each non-core AS receives PCBs representing paths to core AS
each AS has border routers and beacon servers
beacon servers keep receiving beacon messages
every period one PCB sent downstream
finite number of beacon messages
nice scalability with number of links
bad scalability with very dense network topologies

path servers
path servers choose which routes to advertise
core AS operate path servers for down-paths, core-paths
non-core AS operate path servers for up-paths

route creation by host
ask RAINS (like DNS) with URL to resolve
RAINS responds with (ISD X, AS Y, local address z)
ask local path server for path segments (with ISD X, AS Y)
local path servers may asks core path servers
core path servers may ask different ISDs
client receives response with (up-paths, core-paths, down-paths)
combine paths to route to destination

path combinations
each path segment made of single INF and multiple HF fields
combine up, core, down paths to single route
can use peering paths (two AS advertise peering to each other)
can use shortcuts through up and down-paths segments
choose multiple combinations and select best one after testing

optimize for latency, bandwidth, traversed AS

SIBRA
prevented malicious AS to change routes
destination can verify which AS were passed
against DDoS & source address spoofing

8.6 data plane

transparency architecture (predefined routes)
uses determined paths to deliver packets
does path lookup, combines retrieved paths

8.6.1 forwarding

computation & verification of HF MAC value based on local AS secret key
very efficient because only one encryption pass needed (50 cycles)
IP lookup not needed, can simply forward
attacker cannot choose arbitrary own path, would have to guess the MAC

8.6.2 scion packet

common header
Version (4), DstType (6), SrcType (6), TotalLen (16)
HdrLen (8), CurrINF (8), CurrHF (8), NextHdr (8)

source / dest address encoding
DstISD (12), DstAS (20), SrcISD (12), SrcAS (20)
DstHostAddr, SrcHstAddress
addresses can be IPv4+padding or IPv6
constant offset till DstHostAddr

info field
flags (PEER, SHORTCUT, UP), timestamp, ISD identifier, SegLen

hop field
flags (FWD-ONLY, VRFY-ONLINE, CONTINUE, ...)
expiration time relative to info field timestamp
ingress/egress identifiers of target AS
MAC for target AS
AS internally routes from ingress to egress

example
INF's and HF's must be included in packet header (8 bytes per hop)
packet example INF HF HF HF INF HF

reverse route
move the INF fields from start to end of segment

8.7 infrastructure

beacon servers
discover path information
intra-ISD beaconing for core AS to non-core AS
inter-ISD beaconing for core AS to core AS
non-core AS send selected path segments to core

path servers
process path information
store mappings from AS identifiers to path segments

certificate servers
validate path information
keep AS certificates, TRC's
are queried by beacon servers to validate PCB

name servers
resolve domain names to SCION addresses
proposed is the RAINS system

border routers
connect AS together
forward packets to services, internal/other border routers

internal routers
forward inside AS

8.8 deployment

for core AS
manage TRC
sign TRC of neighbouring IDS and endorse them
maintain list of recognised ISD
issue certificates to all AS inside ISD
provide connectivity to neighbouring ISD
generate and broadcast PCB's
provide beacon, name (RAINS), path, certificate, SIBRA & time servers

for normal AS
provide connectivity to other AS
broadcast PCB's
provide beacon, name (RAINS), path, certificate, SIBRA servers

for leaf AS (such as ETH)
obtain AS certificate from core AS
deploy servers (attach PC to border router)
use IP packet till PC reached with existing routers

SIG deployment
simply set up SCION routers on both ends
or map IP prefix to SCION router, opposite the way back

8.9 why it is useful

DDoS impossible
simply not broadcast critical path
attacker cannot bruteforce MAC and therefore cannot reach path
any packets attempting so will simply be dropped
announce hidden/fallback paths to loyal clients if under attack

SDN (Software Definded Networking)
better routing (choose high-bandwidth vs low-latency paths)
better utilization (use multiple paths)
flexibility (split network flows along different paths)
transparency (inter-domain path visibility)
security (application-based secrecy, VPN)
performance (load-balancing, dynamic optimization)

SCION SDN
no prefix hijacking
can prove path traversal
scalability with increased deployment
no single provider lock-in
clean multi-path architecture

8.10 SCION PKI

trust scalability (heterogeneous, no kill switches)
transparency (enumerate trust roots, accountability)
no trust compromise (quick recovery if happened)
trust control (select trust roots)
TLS PKI has transparency with ILS, browsers are trust roots

8.10.1 TRC

defined by ISD, tied to path exploration
efficient update of trust roots
cross signed by CA's, ISDs, core-AS

defines trust roots for
control-plane PKI (core AS certificates)
end-entity PKI (root CA, log server certificates, #threshhold)
name-resolution PKI (root name server certificate)
cross signatures for other TRCs (must not be connected)

fields
description, version, creation, expiration
core AS certificates (control-plane PKI)
root CA certificates (end-entity PKI)
CertLog certificates (end-entity PKI logs)
threshold of signatures required for end-entity certificate
RAINS certificates (name resolution PKI)
quorum TRC, quorum CA
cross signatures from core ASs, ISD's, CA's

update
TRC version number included inside beacon packet
if AS notices new TRC version number will fetch it
potentially very fast change of TRC roots possible

8.10.2 SCION CP-PKI (control-plane PKI)

distribute routes, create routes (BGP + ICMP currently)
needs highest availability possible (because else nothing works)
tradeoff availability / security (need to lookup CA's)

AS certificates
AS name, public key, expiration time
core AS are trust roots
non-core AS obtain short-lived certificate singed by core AS
AS can verify other AS's using cross-signed TRC's
distribution tied to path exploration

host security
host selects TRC (and therefore ISD) it trusts

uses TRC to verify AS certificates

8.10.3 SCION EE-PKI (end-entity PKI)

combine ARPKI (logs) + PoliCert (multiple signed)

subject certificate policy (SCP)
policy that all certificates of domain must follow
contains list of trusted CAs
defines number of signatures required for MSC
defines hard fail or soft fail in case of violated MSC
gets signed by multiple CAs and persisted in public log

multi-signature certificate (MSC)
respects the domains SCP
domain certificate singed by CA's and SCP
confirms that SCP is registered correctly

implications
domain can choose trust roots
need a lot of compromised systems to create bogus certificate

host security
receives SCP / MSC info in same request (server prefetches signatures)
only accepts messages from trusted TRC
other TRC's must provide proof of absence for all trusted TRC

comparison with DNSsec
also contains domain's certificate
but all entities on verification chain must trust
kill switch at each next upper level

8.10.4 SCION NR-PKI (name-resolution PKI)

sign all delegations from resolution process
domain name entry signed by SCP

host implications
DNSsec style used for availability
EE-PKI used for security

8.11 secure control plan messages

secure PCB dissemination (broadcast)
PCB is signed by issuing core AS
each forwarding AS signs it additionally
protect AS specific hop fields with AS specific MAC

failed interface detection
send keep-alive messages to beacon servers
if threshold number of messages is lost, link is declared inactive

secure path revocation
each AS adds revocation token RT to PCB
if router cannot reach host, it sends RT back
host redistributes revocation message to AS path & beacon servers
host constructs new route and tries again

service address
can send packets with service address instead of explicit destination
border routers relay packet to correct service if known
else relay the packet as before

discovery service
each AS has one consistency service
distributed consistent database for active path servers & beacon servers

beacon service
all AS beacon servers connect to consistency service, master service chosen
PCB's are identified by border router, and sent to one beacon server
receiving beacon server will relay message to all other servers
master will register up-path at own path server
master will register down-paths at core path server

path service in non-core AS
all AS path servers connect to consistency service
use this to keep in sync

path service in core AS
non-master path services push / pull from master service
down-path segments are sent to all other core AS

SCMP (like ICMP)
provides network diagnostics like traceroute, and error messages
authenticate with AS certificate or DRkey

8.12 Dynamic Recreatable Key (DRkey)

control-plan protocol, not secure for end-host
prevents source forging on control-plane at very high speed
can build on top of this with DH to secure end-to-end

certificate server (CS)
deployed at AS level
exchange keys with hosts inside AS and other AS
PseudoRandomFunction PRF to derive keys for any entity
assuming entity "Y", AS secret SV_x then key K = PRF("Y", SV_x)
will distribute to routers which can then calculate DRKeys on the fly

first order DRKey
to establish key between AS's
CS_x creates K_x→y with PRF("y", SV_x)
CS_x → CS_y sends (K_x→y) encrypted with y's public key
CS_y creates K_y→x with PRF("X", SV_y)
CS_y → CS_x sends (K_y→x) encrypted with x's public key

second order DRKey
host asks CS for key which responds with DRKey personalized to host
CS can generate key at AS X for host H in AS Y with K_x→y
h = PRF("H", K_x→y))

implications for DNS
can create lookup table with malicious hosts
serve encrypted / verified customers first
use rest of bandwidth for unauthenticated requests

8.13 ISD coordination

addition of new ISD should be possible
but no central entity can control global ISD
no coalition should be able to block valid new ISD
resilience against malicious ISD (fake name, fake GUID, space exhaustion)

main ideas
use PCB to announce new ISD
ISD has GUID and small text description
new ISDs are in 7-day quarantine
detect/resolve conflicts on per ISD basis
provide transparency, instead of consistency

procedure
early announcement is propagated to all ISD, kept for a few days
final announcement only propagated if valid
humans do it on blacklist if invalid

properties
cannot be automated
enables authentication on global scale
conflict resolution done by humans

9 Anonymity

9.1 introduction

targets
hide communication patterns from advisories
hide source from untrusted recipients

applications
criminals (act with impunity)
military (covert attacks/communication/gathering, penetration testing)
trade (prevent price discrimination, hide usage of monitored services)
anonymous reporting (accidents, criminal activity)
human rights (free speech, whistleblowing)
building blocks (bitcoin, ethereum, electronic voting)

archive anonymity
only if all layers are anonymous

9.2 terminology

sender anonymity
sender is unknown
adversary knows receiver and may learn message

receiver anonymity
receiver is unknown
adversary knows sender and may choose message
use a (hidden) service with known pseudonym of unknown receiver

sender /receiver anonymity set
all possible senders, used as rough metric

sender-receiver unlinkability
adversary knows all senders, receivers, but cannot figure out mapping

only works if multiple users
anonymity implies unlinkability

unobservability
adversary cannot tell whether any communication is taking place
archived with DSSS (for wireless) and continuous sending (for wired)
unobservability implies anonymity

threat models
degree of control (local vs global)
type of control (network or compromised infrastructure)
type of behaviour (passive or active)
third party anonymity
compromised destination

9.3 how to send message anonymously

wireless communication
move to avoid triangulation
use burner phone, hacked wifi, ...

proxy / VPN
proxy relays message to correct receiver
proxy has to be fully trusted

not-fully trusted proxy
use layered encryption, any proxy can decrypt single layer
adversary may be able to link input/output because of timing

batching/mixing proxy
proxy collects messages, mixes them, forwards after threshold reached
adversary may collect all in/out per user, do conjunction of receivers

cover traffic with trusted mix
continuous sending of dummy traffic by senders
continuous polling of dummy messages by receivers

mix cascades
dynamic or static (fixed vs changing) number of mixes
original sender prepares return address for each layer

9.4 circuit based systems

performance to enable browsing, but lower anonymity guarantees
layered encryption, but no batching/mixing
establish virtual circuit for flow to use with symmetric keys
lower threat model (local adversary, no confirmation attacks)

terminology
circuit-based anonymous communication systems
commonly known as Onion Routing Systems
main difference to mix-nets is circuit based, onion structure
nodes are called relays (or nodes, routers)
entry (enter network), middle (inside network), exit (exit network) relays

circuit setup
with asymmetric key cryptography
all relays store state of circuit to enable symmetric

direct setup
sender knows all nodes he wants to use in advance
first packet creates state
adversary can trace back if long-term key of relays compromised

telescopic setup
negotiate keys one relay at a time (circuit is extended one node at a time)
layers of encryption look like a telescope
setup is much slower, but nodes do not know if they are the last one
offers forward secrecy (negotiated keys are deleted when circuit is closed)

data forwarding
for all packets of flow (10 minutes for TOR)
AES is used for encryption
sender as established circuit (keys and IDs)
layered encryption, each node removes a layer, forwards to next ID

circuit tear-down
to free state relays and prevent attacks
can be done by sender or intermediate relays
sender tears down starting at furthest away
exit relay tears down if corrupt packet detected

9.5 attacks against TOR

passive traffic analysis
observe edges of network, recoding traffic patterns
flow length, bandwidth pattern, inter-packet timings
real-time or store-compare-later

active traffic analysis
adversary modifies packet timings
delaying, reordering, dropping packets to create observable patterns
flow watermarking (inject marked bit)
flow fingerprinting (inject multiple bits)
website fingerprinting (effective for interactive webpages) ISP observable

prevent traffic analysis
with cover traffic and mixing, introduces significant overhead
only suitable for low bandwidth protocols

prevent higher-layer attacks
per-hop TCP to avoid OS network stack fingerprinting
malware defence

confirmation attack
both exit/entry node from attacker
exit gives resource addressed, entry node give IP of user

sybil attack
adding a lot of identities to skew majority algorithms
publishing a lot of relays in same IP subnet

9.6 TOR

telescopic setup for forward secrecy
per-hop TCP (established on the fly)
per-hop TLS (except last hop)
can support any TCP connection with SOCKS
provides firefox bundle

historical
1981 original email mix-net design
1988 DC-net design
1991 first remailer
1996 original onion design
2004 TOR
2006 TOR project
now most used anonymous communication system I2P
2014 paper is relevant

additional features
end-to-end integrity checking (secure channel between client & exit)
exit policies (exit nodes can restrict destinations they connect to)
multiple streams per circuit (one circuit can reach multiple end users)
censorship resistance (bridges)
hidden services (provides receiver anonymity)

cell
basic unit of TOR, 512 bytes
circuitID 2 bytes, CMD or relay 1 byte, both cleartext
payload 509 byes

command cells
possible commands are create, created, destroy, relay, relay_early
interpreted by receiving cell

relay cells
possible cmds are extend, begin, data, teardown
is decrypted, get fields StreamID, digest, cmd, data
if (digest valid) then execute cmd else replace circuitID & forward

example circuit communication
A → OR1; create c1, E(g^x1)
OR1 → A; created c1, g^y1, H(g^x1y1)
A → OR1; relay c1, {extend, OR_2, E(g^x2)}
OR1 → OR2; create c2, E(g^x2)
OR2 → OR1; created c2, g^y2, E(g^x2y2)
OR1 → A; relay c1, {extend, g^y2, H(g^x2y2)}
can now start to send useful stuff
A → OR1; relay c1, {{begin bob;80}}
OR1 → OR2; relay c2, {begin bob;80}

why relay_early
if able to build circular flow then amplification trivial
no maximum path length therefore allows cheap DoS
relay_early caps max circuit length at 8
relay will throw error if relay command is received with extends request

hidden services
hash of public key is identifier of hidden service (kawkjbfekjsafkj.onion)
introduction point IP connects services and users
they establish a rendezvous IP, and meet there

facebook hidden service

to avoid that a lot of connections reach fb from a few IPs (tor exit nodes)

directory authorities DA
track state of relays, store their public keys
also act as bandwidth authorities (verify bandwidth of relays)
limit the number of relays per IP subnet (counter sybil)
run consensus algorithm, new document signed every hour
client has list of authorities keys, accepts document if signed by > 50%
10 total in TOR, only 5 have to be compromised

relays
have to periodically report signed statement of state and statistics
high load relays are verified by DA (acting as bandwidth authorities)
act as directory cache (to avoid scalability issues)

censorship resistance
not publicly listed bridge relays distributed personally
but can deep-inspect packet to knows its a Tor packet
to disallow packet inspection can obfuscate traffic with pluggable transport

9.7 decoy routing

network based censorship evasion mechanism
ask trusted router for decoy location
router then redirects instead to censored place
routers know real location from abused TCP fields (like sequence number)

registration
alice gets public key of router g^y from somewhere
established shared key using the covert channel (sequence number)
needs around 10 TCP setups till router can register client

deflected communication
router detects incoming connections from a registered clients
closes initial TCP to decoy
opens new secure connection with Alice
Alice sends real address to router

issues and limitations
keep state for each client
perform asymmetric cryptography (censor can DoS)
traffic patterns are detectable
no deployment incentives

9.8 network layer anonymous communication

9.8.1 lightweight anonymity and privacy (LAP)

first packet builds up header
bit determining which way, index determining offset
ASes add own per-hop information, encrypted & MAC
AS on path replaces source with self (need to trust first AS)
and adds per-hop routing information to header (encrypted, verified)
attach this header to all data packets, AS simply decrypt & forward

properties
lightweight packet processing (no asymmetric crypto)
no detours, use same path as normal traffic
limited amount of state on routes (packet-carried state)

pros
very lightweight, only symmetric crypto, stateless forwarding
same path as normal packets, therefore fast
simple (easy to reason about security, implementation)

cons
very limited threat model (full trust in own ISP)
clients need to be protocol aware
header causes bandwidth overhead
no payload encryption

9.8.2 source accountability w/ domain-brokered privacy (APNA)

anonymity is problematic for ISP (because need accountability)
tries to strike balance between privacy and accountability
ISP acts as trusted third party
ISP can attribute every packet to sender
but then hides host identify outside ISP
facilitates end-to-end encryption

properties
issues temporary ephemeral IDs (EphID) to customers
linked to customer identity by ISP
leaks only ISP identity to other entities
privacy preserving
forward efficiency (low computation, low storage overhead)

host bootstrapping
establish host identity HID
establish shared key K_ha with ISP
ISP has symmetric secret key K_s, public/private keypair K_isp

EphID setup
host creates K+,K-, sends K+ to EphID server
EphID = K_s(HID | ExpTime), C_ephid = (K+, EphID}_K_isp-
server sends back C_ephid to host

connection establishment
host A learn EphID and C_ephid from host B
A and B compute secret shared key K_ab using the certificates

data communication
host to ISP with SrcEphID, DstEphID, payload, MAC using K_ha
ISP can compute HID with K_s from SrcEphID
ISP looks up if MAC correctly created with K_isp
ISP relays packet to other ISP
other ISP resolves target host using the EphID

9.8.3 high-speed onion routing at network layer (HORNET)

choose path & obtain public keys of all routers on path
setup phase; hosts establish a shared key with all routers on path
data phase; headers allow routers to extract routing information
typically one header for forward, one for backward

properties
anonymity as a service (similar to LAP)
full onion routing at network layer
stronger threat model than LAP (similar to Tor)
tunnel setup with asymmetric crypto
fast data forwarding with layered payload encryption
needs path aware routing

requirements
stateless forwarding (routing infos in header)
low latency (same path as normal packets)
asymmetric path support (contrary to LAP)

communication overview
needs path-aware routing (like in SCION)
sender obtains all public keys on route
construct two headers, one for forward, the other one for back

path information hiding
encrypt & MAC after appending forwarding information
keep payload same size by cutting of payload the same size as forward info

pros
support for strong threat model
low processing / storage overhead

cons
large header (bandwidth overhead)
setup is bottleneck (because asymmetric crypto)
client & destination both need to participate
no deployment incentives
limited anonymity if destination is close

10 firewalls, intrusion detection, evasion

10.1 firewall

system used to protect trusted network, filters ingress/egress traffic
configured policies device, can permit, deny, proxy data

filtering types
ingress (filter incoming traffic, low to high security)
egress (filter outgoing traffic, high to low security)

default action for unknown traffic
allow (allow if no rule found which denies it)
deny (disallow if no rule found allowing it)

deny access techniques
drop (simply "lose" the package)
deny (inform source)

organisational challenges
ruleset management (huge & complicated rules, grown over time)
operations vs security teams (connectivity vs security)

challenges
unable to inspect encrypted traffic
high number of false positives

packet capturing at high speeds (inspection is expensive)
latency introduced by inspection engines
application level attacks (CSS)
policy, signature management

accuracy vs precision
precise if measurements are very close to specific value
accurate if specific value is correct
accuracy = (true positives + true negatives) / total

detection tradeoffs
can detect almost everything, but runtime suffers
choose tradeoff between security and speed
high number of false positive very bad
cutting the wire is perfectly secure but useless
rate of false positives very important
combine/chain tests to increase precision

10.2 firewall types

10.2.1 network firewall

protects different networks
software on specific hardware
virtual machine which filters traffic between two or more networks

10.2.2 host firewall

protects single host
layer of software that controls in/out network traffic of specific machine

10.2.3 stateless firewalls

examine a packet at network layer, decision based on packet header

pros
application independent, good performance, scalability

cons
no state, no application context

10.2.4 stateful firewall

keep track of state of network connections, decision also based on network

pros
more powerful rules

cons
UDP has no state, host vs firewall inconsistent states, state explosion

10.2.5 next generation firewall NGFW

deep packet inspection, depending on protocol and application
take application & protocol state into account for security decision
block packets which do not fulfil protocol specification
multiple firewall devices share state

problems with simple firewalls
access control enforcing is not enough
firewalls can't block all malicious traffic
peer-top-peer, stupid users, multiple applications use same ports

pros
application & protocol aware

cons
need to support a lot of applications
performance & scalability
host vs firewall inconsistent states

10.2.6 web application firewall WAF

firewall sits before web application, and filters requests

request filtering
request patterns
SQL injection, CSS, buffer overflow attempts, checking POST parameters
static/dynamic blacklisting / whitelisting

pros
faster reaction times compared to security updates for application

cons
false positive problem

10.3 detection

10.3.1 evolution

signature based detection (multi-level)
machine learning, classification mechanism

10.3.2 types

reactive, static
detects known attacks using signatures
can not react to new attacks

proactive, dynamic
system detects known and so far unknown attacks
higher chance of false positives

deterministic
same input produces same output
uses blacklists, signatures, rules

non-deterministic
system detection is fuzzy (heuristics, learning, sandbox)
reason for block often not known

10.3.3 intrusion / attack detection methods

protocol analysis
analysis & decoding of protocols
normalization of traffic for the specific protocol

signatures
compare attributes using white / blacklist
compare attributes using pattern matching

sandboxing
execute suspicious file in sandbox
actions are categorized and evaluated

machine learning
make decisions based on data & assumptions formed from previous data

10.3.4 signature based detection

each threat has signature, crafted by security researcher
sophistication depends on human implementation
frequent updates of signature database necessary
identifies & labels threat very fast
almost no false positives, deterministic, but reactive

10.3.4.1 types

one-dimensional
indicate known good, known bad (from white/black lists)
very fast & efficient, low false positives
but reactive, humans, frequent updates needed

two-dimensional
regex string / function matching, search binary strings
more flexibility than one-dimensional

multidimensional
signatures triggered when specific actions are executed
actions are classified as good/bad
if threshold reached, the threat is classified

10.3.5 sandboxing

run malware in sandbox environment
use machine learning to classify behaviour
apply classification to behaviours (good or bad) and detect malware

pros
proactive, no signature updates

cons
resource intensive, high latency, difficult to scale
need exact replica of target
attacker can simply pause malware for the first hour

10.3.6 machine learning

supervised learning
feed good/bad samples
compares all behaviours, and assigns weights to conclude a result

unsupervised learning
no examples needed, simply responds with function
when baseline is reached threat is classified somehow
proactive, but do not know why it is blocked

10.4 attack methods

vulnerabilities of used software
exploit OS / firewall bugs
DoS with state explosion
using fallback policy to advantage

source spoofing
IP source spoofing (only UDP)
using vpn inside network (abuse DNS/ICMP)

fragmentation
attack send by multiple packets, doubling packets, out of sequence
port only in first packet, filter ignores subsequent packets

encoding
URl encoding (linux vs windows difference)
HTML obfuscation (different encodings, compression, insert noise)
chaffing in invalid chars decoder on client removes

10.5 malware deployment cycle

develop new malware
develop malware with key functionality
crimeware (exploit kits like Metasploit, Phoenix, Eleonore)

serial variants / permutations
churning out new versions of the malware which do the same thing
release new batch as soon as old one is detected

crypters
encrypt malware such that static code analysis & signature become useless
only decrypt code about to be executed
examples include ETCV by Topcat_42, Public Crypter Poison Ivy

protectors
detects use of debuggers/sandboxes
tries to debug itself, watches out for specific flags
examples include code virtualizer, rdgsoft

packers
make binary kits smaller & more portable
changes order, swaps constructs, inserts noise, sets compiler flags
detected using tools such as Exeinfo PE and PEID

binders
embed trojan code into legitimate files
examples include CrimeBind, PE-Protect v2

quality assurance
do hardening (test the batch with commercial antivirus)

10.6 layered defence

perimeter
ISP, firewall
servers, desktops protected by this

host
anti-virus, browser
desktops, laptops protected by this

10.7 attack types

direct attack
attacker instantiated
server-side exploits
attack running program on server

indirect attack
target instantiated
the user executes a program he shouldn't have
attacker has little control over when and how it happens

11 Broadcast Authentication and Stream Signatures

11.1 setting

communication assumptions
sender uses broadcast to disseminate data
receives user unicast communication channel
broadcast channel is lossy

adversary model
eavesdrops all messages
injects messages
can be a receiver
but sender is assumed to be trusted

target security metrics
external adversary can not forge a message
receiver / group of receivers can not forge a message

efficiency metrics
communication / computation / storage overhead
delay for authentication, signature
resilience to packet loss

security / efficiency
scalability
handle dynamic membership (join / leave)

11.2 signatures remarks

signature problems
can be attacked with signature flooding attacks
transmission overhead due to larger message
one signature over multiple packages is intolerant to packet loss

asymmetry by time
sender publishes authentic commitment F(K) with message at time t
key k disclosed at t' > t
receiver can verify with F that message is authentic

11.3 single MAC broadcast authentication

sender attaches single MAC to each packet
receiver obtains MAC key K

good
low computation, low latency, no extra storage, good scalability
perfect resilience to packet loss
good for dynamic membership
secure against external adversary

bad
any recipient can generate a message which the key K
only works if all receivers are fully trusted

11.4 Multi MAC scheme

generalization of single MAC scheme
large number of MAC keys, each member gets subset
for each key a MAC is computed and sent with the message
receiver checks all MACs where the key is known

good
low computation overhead, no buffering, verification delay
robust to packet loss

bad
several bad receivers can forge message
not scalable (high communication overhead)
overhead for dynamic membership

11.5 small RSA signs

use short-lived small RSA keys (fast to generate & verify)
periodically send out new public key signed with strong signature

good
low computation / storage overhead
no buffering, verification delay
scalable

bad
high communication overhead (> 50bytes/packet)
time synchronization needed (so old keys no longer accepted)
does not provide non-repudiation, long term security
not very robust to packet loss

11.6 time efficient stream loss-tolerant authentication (TESLA)

each packet has key from two intervals before, payload, encrypted mac
can verify all packets which were received earlier than two intervals

needs security condition
receiver knows key disclosure schedule (need time synchronization)
on arrival of packet receiver knows sender did not yet disclose its key
else drop packet

how to bootstrap receivers
loose time sync (upper bound on senders time using a max time sync error)
session setup (start time, interval duration, disclosure delay)
digital signature for initial authentication

use one-way hash chains
prepare long hash chain
limited size of hash chain
each second for 32 years needs 2ˆ30 chain entries, takes 12 minutes

store logn entries

good
receivers can only verify
provides signature property (if time stamping reliable)
low communication, computation overhead
packet loss no issue, scalability
no impact on dynamic membership

bad
storage required for old messages which are not authenticated yet
requires time synchronization
delayed authentication

applications
authentic media broadcast
galileo message authentication (time synchronization after 12h)
IoT networks
secure routing protocols
time synchronization with multiple TESLA instances

11.7 one-time signature

make signatures cheaper
amortize single digital signature to sign multiple messages
constructed with efficient cryptographic hash functions
efficient for generation & verification
publish / connect stuff with public key

lamports one time signature
for each bit have two private/public key pair

improved lamports construction
only one keypair for each bit
publish checksum bits which encore # of signature bits were 0

improved lamports construction 2
goal is to reduce size of signature
use two hash chains; one for signature, one for checksum
S0 → S1 → S2 → P
C2 → C1 → C0 → P
publish pair (S_i, C_i)

merkle-winternitz construction
do improved construction 2 in parallel
each two bits have chain (S0 → S1 → S2 → P, S'0 → S'1 → S'2 → P')
checksum chains encode sum (Si, S'j) → i+j

applications
sign low-entropy data
quantum resilient
high-speed command-and-control messages (non-repudiation)

11.8 stream signatures

first packet is authenticated conventionally
each packet n contains authentication for the following packet n+1

Gennaro & Rohatgi off-line
sender knows entire stream before sending
last packet has no hash, only D_n
packet contains (D_i, H_i) for H_i = H(D_i+1, H_i+1)

Gennaro & Rohatgi on-line
sender prepares authentication for next packet
first packet has no payload, but bootstraps pk_0
packet contains (D_i, sk_i, pk_i+1) for sk_i = (D_i, pk_i+1)
so each sk_i can be verified with pk_i from packet before

advantages
secure against malicious clients
low computation & communication overhead
tolerates packet injection
no extra storage (online)
no delay for packet verification
dynamic membership

disadvantages
can't deal with loss
online has high communication overhead
sender has to save entire stream with offline approach

11.9 streamed merkle hash tree

group stream into s^d packets
for each group build tree
each packet contains singed root and nodes to hash to root
verifier can verify all packets

advantages
secure against malicious clients
low computation, storage overhead
tolerates packet injection
packet loss robust
dynamic membership

disadvantages
buffering & delay at sender
high communication overhead

tradeoff
sender buffering & verification overhead
signature generation & verification overhead
balance by choosing appropriate size 2^d

11.10 signature flood attack

send a lot of stupid signatures receiver needs to process

countermeasures with TESLA
use TESLA to authenticate origin, if valid proceed to check signature
as TESLA requires time sync & late authorization need to save the packets
to minimize storage needed first only receive MAC
if MAC is validated later, accept corresponding packet content
drop any other packets

countermeasures with merkle
bind together data to single signature
only verify signature if enough packets received which resolve to it
attacker has to send a lot of packets until signature is verified

11.11 VANET

network of cars which send messages
based on beaconing

RHCN
road hazard condition notification
inform cars about aquaplaning, iceing, ...

EEBL
emergency electronic brake light
when someone breaks abruptly a warning is send

CCW
vehicle know each others positions
helps to navigate, prevents accidents

IEEE 1609.2
standard which defines cryptographic algorithms for car communication
needs non-repudiation, so each message needs signature
ECDSA (elliptic curve digital signature algorithm)
small message size (only 64byte)
signature generation faster than verification (1 to 4)
messages sent out with 10hz, 5 senders already overload computation
therefore only verify message if content would be critical
but attacker can simply create messages which would be critical

12 probabilistic traffic monitoring

trade accuracy for efficiency
measure with limited power, but accurate output with high probability
router summarizes traffic to compact dataset, sends it to statistical server
focus on traffic properties (large flows, number of flows, accesses)
can add second stage which monitors the flagged flows accurately

12.1 traffic flow

sequence of packets, where some header fields have pattern

flow identifier
5 tuple (Src IP, Dst IP, Src port, Dst port, protocol)
detect DDos (∗, Dst IP, ∗, ∗, ∗)

per flow monitoring
keep counter for each flow
RAM is very expensive; attacker can overflow your device
average packet size 200bites, 200'000 flows per 10Gb/s

probabilistic traffic monitoring

large flows
flows that consume more than given threshold of capacity
small fraction of flows cause large portion of traffic

accuracy metrics
false positive (include small flow)
false negative (fail to identify large flow)
measurement error (computed traffic volume wrong)

applications
anomaly detection (enforce quality of service, sending rate
network management (accounting, load balancing)

remaining challenges
measurements increase risk of DDoS
requires source authentication
attacker can frame others if he knows the hash function being used

12.2 Sampled NetFlow

samples one of k packets to estimate traffic
persist the 5-tuple as key
for each key persist # of packages + total observed size
at the end of sampling, multiply results with k

advantages
simple, reduced processing time

disadvantages
memory overhead
short lived flows inaccurate
may over-count (up to k),
may underestimates
false negatives
non-random sampling & biases small packets

12.3 sample and hold

check for every packet if flow record exists
if yes, hold the packet
if no, sample with probability relative to packet size
keep record of sizes, packets of flows

advantages
no overcharging
no false positives

disadvantages
inspects headers of all packets
false negatives

12.4 multistage filter

hash 5-tuple to [1,...,#counters]
add packet size to result
warn if threshold is reached

multiple stages
make it better by using multiple counters
each counter has own independent hash function
to estimate flow size, take min() of all values reached by hash functions

nice hash function
encrypt ID with secret key using AES, which gives 128 bits
so can use 20bits for each stage as hash function

abusing AES hash
if attacker knows key, can make specific flow appear malicious
can create multiple flows such that each flow collides in one bucket

advantages
no false negatives
fixed memory resources

disadvantages
false positives because collisions

12.5 counting algorithms

select random item from unknown size n
change selection with probability 1/i where i # of passed items

majority algorithm
persist item_type & counter
increase counter if item same
decrease counter if not item
take new item_type if counter is 0
if some item has more than 50%, then is item_type persisted at end
but need to pass again, because if no item >50%, item_type will be random

MG algorithm (more-than-k)
needs n=m/k - 1 channels where m is input size
if item already in channel increase counter
if not & no slot free, decrease counter of all, if counter 0 remove it
if not & slot free, put it in slot

EARDet Algorithm (more-than-k-size)
based on MG
take packet size into account, not number
define threshold, if reached, flow is put on blacklist
(flows on blacklist are counted exactly)
diminish all counters in specific time period
add virtual flows for idle periods
no false positives for small flows
no false negatives for big flows
deterministic, small storage needed
can detect bursts of traffic

12.6 finding duplicates

bloom filter
create bitvector with m elements (m = big)
hash element with k hash function giving k positions
if all k positions are one, element has been seen before, else set all to one
has false positives (can influence probability by parameters m&k)
no false negatives
very time & space efficient

bloom filter over time
because can not remove elements
can use two filters in parallel
each time slot one filter is cleaned and then filled up
both filters are checked for collisions
stabilize by including max delay time in timeslot length

12.7 estimating number of different flows

naive recording
infeasible, O(n) storage space

increase counter if new packet arrives
use bloom filter to see if packet is new
still O(n) memory overhead

use hash
each packet is hashed, only lowest hash value is persisted
can now estimate number of flows by 1/hash
only O(1) storage

use k-hash
to be more robust, generalize to persist the k smallest (reduce variance)
can now estimate number of flows by k/hash_k

13 Guesttalks

13.1 email filtering

checks
dns domain queries (detect non-existent sender domains)
dns dob queries (detect newly-registered sender domains)
dns spf & dkim queries (detect forged sender addresses)
dns blacklists (block spam hosts & botnets)
dns blacklists with a high refresh rates (block spam host ip changes)
smtp-time recipient address verification (block "joe jobs")
smtp helo checks (detect spammers & botnets)
content checks (clamav& spamassassin)
manual inspection queue (suspected phishing & malware)

phish trap
messages land in queue, need to be manually released
privacy, performance issues, 1.5 mio messages per year

overall old system
needs anti-virus, linux upgrade
is inaccurate, needs too much manual work
is slow (users complain)

migrate to MailCleaner
web portal with personal quarantine
can set personal preferences, process own quarantine

good at MailCleaner
mailcleaner disarms critical attachments
has blacklists, does DNS checks
users can send mail to servicedesk for attachments
users can send mail to mailcleaner to report spam

bad at MailCleaner
no mail-client plugin to report spam
spam rarely reported
any recipient of list can release message for all

criminal activities
identity theft (gain access to accounts)
extortion (ransomware, blackmail, DDoS)
spyware (espionage)
botnet (use computer for mail, DDoS attacks)
cryptojacking (use computer for mining)
scam (persuade to pay bill)

phished account usages
send spam/phis/malware mails
set up phishing webpages

evil messages difficult to catch
invitation to click a link or open an attachment
ups shipping labels / order confirmation
invoices, documents to view/sign
minimal text / images + URL

phishing countermeasures
limit message number send by user
update filters frequently
remove phishing mails from user mailboxes
lock accounts
neutralize phishing URLs (redirect to warning webpage)

future mail filter techniques
combine records of inbound & outbound mail
sandboxing, web proxy & automatic updates of URL blacklists
blacklists of fake journals & conferences
smarter mail-clients, browsers & operating systems

13.2 Adups FOTA mobile backdoor

over the air firmware updater
sends unencrypted device data
contains IMEI, IMSI, even SMS, mobile status (allows to localize)