Chair for Network Architectures and Services
Department of Informatics
TU München – Prof. Carle

Network Security

Chapter 14

DNS Security, System Security, etc.

Part I: DNS Security

- Part I: DNS Security

History and Motivation of DNS

- Problem: The Internet needs IP addresses. Human beings do not memorize IP addresses well.

- Idea: Map easy to remember symbolic names to IP address
  - www.net.in.tum.de → 131.159.15.231

- (Not so good) first approach: hosts.txt
  - Local file on every (!) machine
  - Updates needed to be applied manually(!)

  → Feasible for small networks,
  not feasible for the internet
- Better approach: Domain Name System (DNS)
  - Paul Mockapetris, 1983
  - Wide deployment on the Internet starting 1988
  - RFCs: 1034, 1035

Picture: http://www3.isi.edu

Overview

- DNS is a distributed name database
  - Worldwide deployment
  - Hierarchic structure
  - DNS Names are unique

- DNS is a protocol on Application Layer
  - Resolves symbolic names to IP addresses
  - Operating system provides a stub resolver and needed system calls "`getHostByName`"

- DNS is extensible, e.g.:
  - ENUM (Telephone Number Mapping): +4989…123 → voip.example.com
  - DNSSec (DNS Security Extensions), covered later in this lecture

Terminology:

- Zone ~ administrative unit within the DNS
- A Zone‘s nameserver saves information in a Zone File
- A Zone File consists of several Resource Records (RR)
  - Example: `foo.org. 3600 IN A 12.34.56.78`

- The RR can be split into the following fields
  - Owner
    - In case of A RR: DNS name
  - TTL (Time to live)
    - Validity of a cache entry in seconds (optional)
  - Class
    - „IN“ is in use today only
  - Type
    - Type of RR
  - RDATA
    - In case of A RR: IP to be mapped on DNS Name

Most important Resource Record Types

Typ	Description
A	Mapping Name → IPv4 **A**ddress `foo.org. 3600 IN A 12.34.56.78`
AAAA	Mapping Name → IPv6 **A**ddress `foo.org. 3600 IN AAAA 2001:db8::1`
MX	Name of the mail server (**M**ail **Ex**changer) of the domain foo.org `foo.org. 3600 IN MX 10 mail.foo.org.`
NS	**N**ame**s**erver of a domain `foo.org. 1800 IN NS ns.foo.org.` `ns.foo.org 1800 IN A 12.34.56.79` („Glue Record“)
CNAME	Alias name for a A resource record (**C**anonical **Name**) `www.foo.org. 3600 IN CNAME foo.org.`
PTR	Mapping IP address to name (**P**oin**t**e**r**) `78.56.34.12.in-addr.arpa. 3600 IN PTR foo.org.`
	Many more: CERT, TXT, ISDN, SOA, etc…

DNS Packet Layout

		12	variabel	variabel	variabel	variabel	[byte]
IP	UDP	DNS-Header	Query	Answer RRs	Authority-RRs	Additional RRs	

- DNS uses UDP
  - efficient, no connection establishment needed like with TCP
- DNS-Header:
  - message ID, amount of entries in the following fields, further control information (e.g. for recursive/iterative resolving), authority bit , …
- Queries:
  - Specifies the query: DNS name, RR Type, RR Class
  - E.g. www.foo.org IN A
- Answer-RRs
  - One or several Resource Records with the requested information
- Authority/ Additional RR:
  - name(s) of the authoritative nameserver(s) for this query

DNS Packet (Example from RFC)

Query:

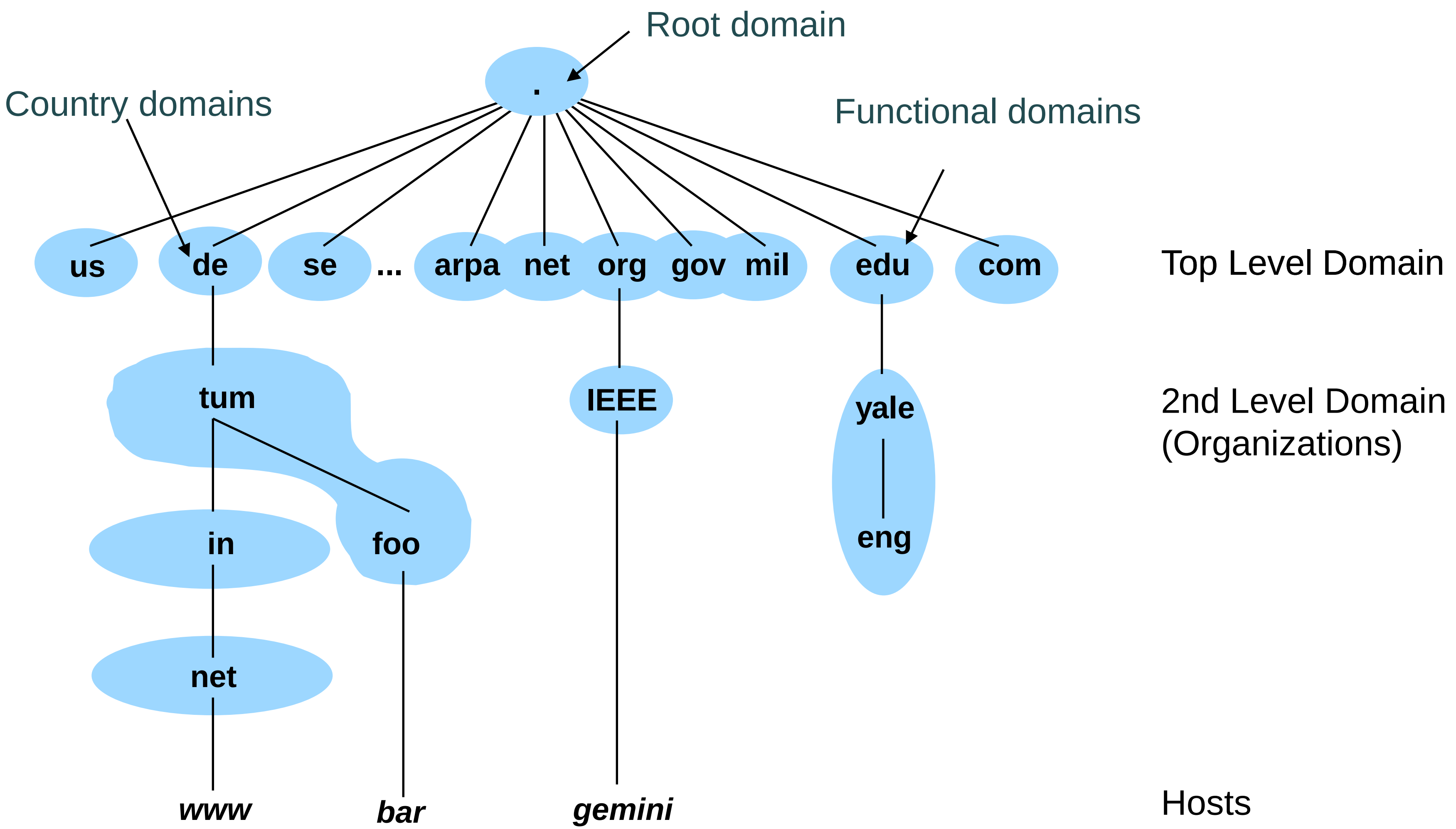
```
            +---------------------------------------------------+
Header      | OPCODE=SQUERY                                     |
            +---------------------------------------------------+
Question    | QNAME=SRI-NIC.ARPA., QCLASS=IN, QTYPE=A           |
            +---------------------------------------------------+
Answer      | <empty>                                           |
            +---------------------------------------------------+
Authority   | <empty>                                           |
            +---------------------------------------------------+
Additional  | <empty>                                           |
            +---------------------------------------------------+
```

Response:

```
            +---------------------------------------------------+
Header      | OPCODE=SQUERY, RESPONSE, AA                       |
            +---------------------------------------------------+
Question    | QNAME=SRI-NIC.ARPA., QCLASS=IN, QTYPE=A           |
            +---------------------------------------------------+
Answer      | SRI-NIC.ARPA. 86400 IN A 26.0.0.73                |
            |               86400 IN A 10.0.0.51                |
            +---------------------------------------------------+
Authority   | <empty>                                           |
            +---------------------------------------------------+
Additional  | <empty>                                           |
            +---------------------------------------------------+
```

DNS Name Space

- The name space is hierarchically structured into zones
- One zone corresponds to a subtree of the DNS Name Space

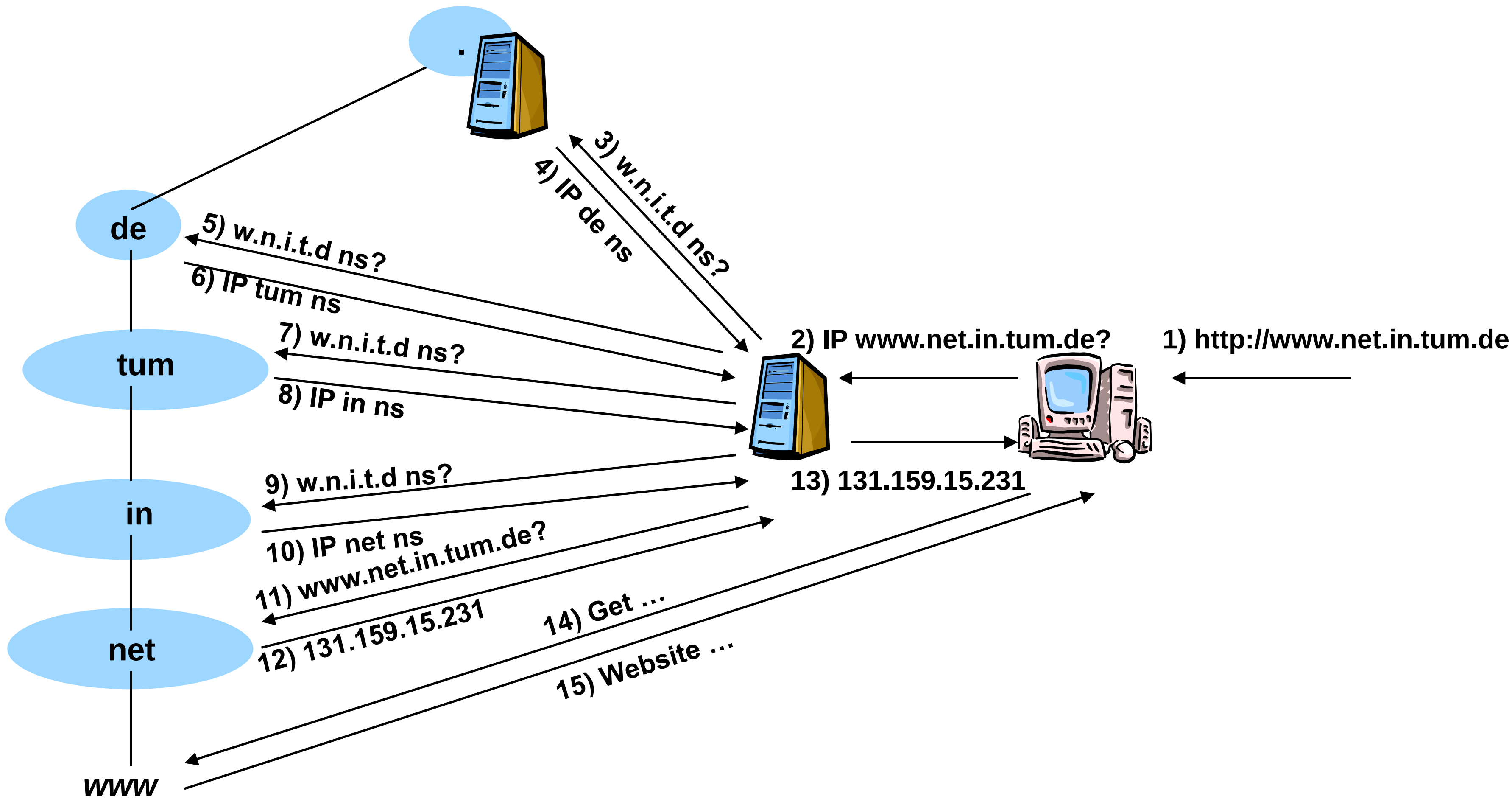


Name Server (NS)

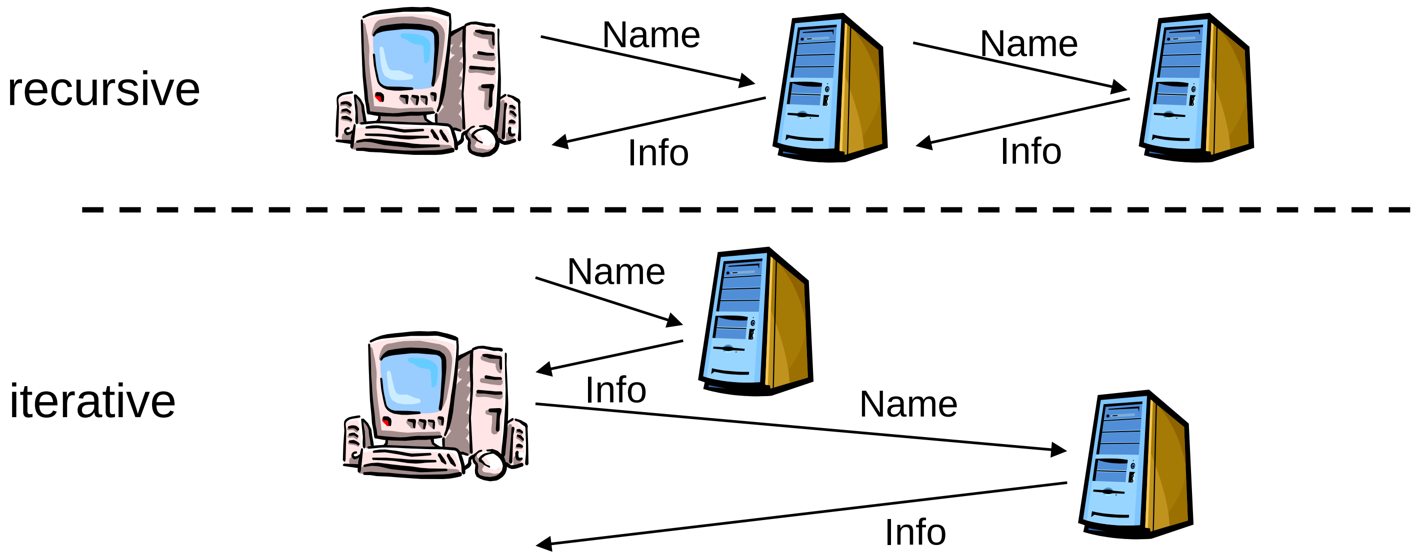
- Each zone has one primary and 0..n secondary name servers
  - Every NS only knows a part of the DNS name space
  - Every NS only knows the IP addresses of „his“ nodes and the NS of „his“ subdomains.
  - Every NS caches DNS responses
  - Secondary NS are updated against the primary NS („zone transfer“)

- NS are also queried by stub resolvers (“hosts”) for DNS lookups

Iterative Name Lookup (Example)



Iterative vs. Recursive DNS Lookup



DNS and Network Security

- DNS was designed at a point in time, where security was no issue due to the small amount of network users (mostly scientists).

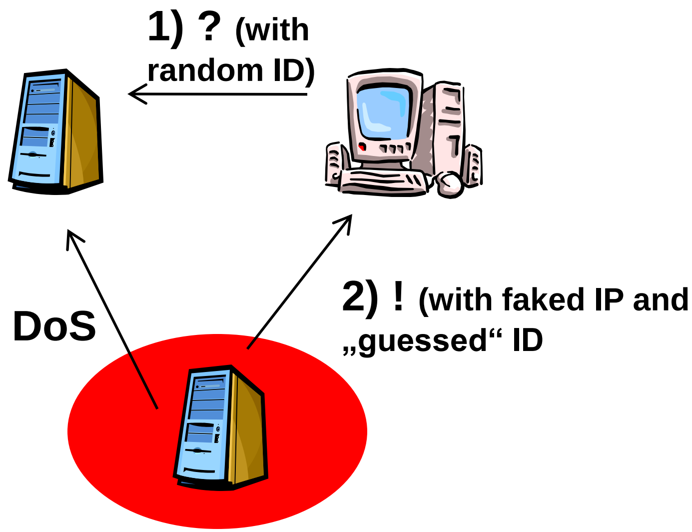
- Security was neglected in DNS.
  - DNS Responses are not signed
  - Receiver of DNS responses cannot validate the authenticity

- Possible impact of successful DNS hacks:
  - Updates for the OS are downloaded from a fake server
  - Users log into fake websites
  - Mails are delivered to fake mail servers
  - etc…

➔ The security of the internet depends on the security of DNS

Attacks on DNS (simplified)

- Examples for attacks
  - „Answer with a fake response before the legitimate DNS server does“



  - DNS cache poisoning: use an exploit inside the DNS software for adding faked entries to the DNS caches
  - „Kaminsky attack“ (2008): „Become DNS server for every zone you like!“
    - Severe attack! Kaminsky decided not to publish the attack but warned DNS software manufacturers about the attack
    - DNS software got patched worldwide
    - Finally Kaminsky dared to publish the attack!

DNS Security Extensions - Basics

- Privacy of DNS queries/replies is no goal

- Basic idea: make DNS safe using digital signatures
  - Safety here means: “Make sure that DNS replies are valid”
  - Can be achieved by signing RR of a zone.

- Digital signatures are based upon public key cryptography
- Private Key (only known by the owner of the zone) signs data
- Public Key (made public) is used for validation of signatures

- Basic question:
- Where to take the public key from to validate a signature?
- How to make sure, that a public key is “valid”, i.e. really belongs to a certain entity?

➔ Use a Chain of Trust

DNSSec Chain of Trust

- DNS servers obtain public/private keys
  - Only the public keys of the root servers need to be well known, are e.g. „built-in“ the operating system (like webbrowser‘s cert store)

- Root servers sign (using their private key):
  - All RRs of the Root zone (e.g. NS RRs of all TLDs, e.g. „.de.“)
  - The public keys of the owners of the TLDs using DS RR (Delegation Signer) → Root servers vouch for the validity of the TLD‘s public key.

- Chain of trust continues: TLDs sign (using their private keys):
  - All RR‘s of their zone (e.g. „.tum.de.“)
  - The public keys of the owners of the Second Level Domains

- (Analogous for deeper hierarchy levels, e.g. “in.tum.de”)

➔ A chain of trust is established from root servers down to subdomains

New DNSSec Ressource Records

Typ	Beschreibung
DS	The „parent zone“ publishes the fingerprint of the public key used within her „child zone“ (**D**elegation **S**igner), e.g. the root server have a DS RR for „.de.“ `dskey.example.com. 86400 IN DS 60485 5 1 ( 2BB183AF5F22588179A53B0A 98631FAD1A292118 )`
RRSIG	**Sig**nature over all records within a zone with the same owner, type and class, e.g. all A RRs of class IN for host.example.com `host.example.com. 86400 IN RRSIG A 5 3 86400 20030322173103 ( 20030220173103 2642 example.com. oJB1W6WNGv+ldvQ3WDG0MQkg5IEhjRip8WTr PYGv07h108dUKGMeDPKijVCHX3DDKdfb+v6o B9wfuh3DTJXUAfI/M0zmO/zz8bW0Rznl8O3t GNazPwQKkRN20XPXV6nwwfoXmJQbsLNrLfkG J5D6fwFm8nN+6pBzeDQfsS3Ap3o= )`
DNSKEY	Contains the public key that can be used to verify signatures within a zone `example.com. 86400 IN DNSKEY 256 3 5 ( AQPSKmynfzW4kyBv015MUG2DeIQ3 Cbl+BBZH4b/0PY1kxkmvHjcZc8no kfzj31GajIQKY+5CptLr3buXA10h WqTkF7H6RfoRqXQeogmMHfpftf6z Mv1LyBUgia7za6ZEzOJBOztyvhjL 742iU/TpPSEDhm2SNKLijfUppn1U aNvv4w== )`
NSEC, NSEC3	Contains the name (hash value) of the lexicographically following DNS name `alfa.example.com. 86400 IN NSEC host.example.com.`

NSEC RR (1)

- Question: How can one believe in a „negative" query response?
  - E.g.: „Response: There is no DNS name `b.foo.com`".
  - An attacker could have sent this to deny the existence of `b.foo.com`

- Approach: use „authenticated denial of existence" (NSEC)
  - Sort domain names alphabetically,
  - concatenate the sorted domain names cyclically with NSEC RRs,
  - sign NSEC RR (using RRSIG-Records)
  - Example: foo.org has: alpha.foo.org, beta.foo.org and gamma.foo.org

```
alpha.foo.com. 86400 IN NSEC beta.foo.com. ( … )
beta.foo.com. 86400 IN NSEC cesar.foo.com. ( … )
cesar.foo.com. 86400 IN NSEC alpha.foo.com. ( … )
```

- Note: This list can be precomputed. I.e. the server does not need to compute a special message to deny the existence of a subdomain. Decreases CPU load on the nameserver.

NSEC RR (2) - Example

```
alpha.foo.com. 86400 IN NSEC beta.foo.com. ( … )
beta.foo.com. 86400 IN NSEC cesar.foo.com. ( … )
cesar.foo.com. 86400 IN NSEC alpha.foo.com. ( … )
```

- A query for the A RR of **b**.`foo.com` will be answered with: **alpha**.`foo.com. 86400 IN NSEC` **beta**.`foo.com. ( … )` including the signature.
- The resolver validates the signature and evaluates the massage:
  - „The subdomain next to **alpha**.`foo.com` is **beta**.`foo.com`"
  - ➔ There is no **b**.`foo.com`!

➔ The resolver can be confident, that **b**.`foo.com` really does not exist.

Zone Walking

- Problem: NSEC RR can be abused to enumerate all DNS entries within a zone ("Zone Walking").

- The attacker only needs to send enough well chosen queries for DNS names, e.g.:

  Query for host „**`b`**". Response: **`alpha`**`.foo.com NSEC `**`beta`**`.foo.com`

  Query for host „**`c`**". Response: **`beta`**`.foo.com. NSEC `**`cesar`**`.foo.com`

  Query for host „**`a`**". Response: **`cesar`**`.foo.com NSEC `**`alpha`**`.foo.com`

- The attacker finally knows all subdomains **`alpha`**, **`beta`**, and **`cesar`**.

- Privacy concerns!
  - Zone walking was of the most important reasons, that prevented the quick deployment of DNSSec.

NSEC3 RR (1)

- Hashed Authenticated Denial of Existence (NSEC3)
  - Hash all host names with a well known algorithm,
  - sort the hash values in alphabetical order,
  - concatenate the sorted domain names cyclically with NSEC RRs,

```
177d..7f7e 86400 IN NSEC3 857a..af32 ( … )
857a..af32 86400 IN NSEC3 a25c..a018 ( … )
a25c..a018 86400 IN NSEC3 177d..7f7e ( … )
```

  - sign NSEC3-RRs.

NSEC3 RR (2) - Example

```
177d..7f7e 86400 IN NSEC3 857a..af32 ( … )
857a..af32 86400 IN NSEC3 a25c..a018 ( … )
a25c..a018 86400 IN NSEC3 177d..7f7e ( … )
```

- Query for host „**`b`**“ is received by DNS server.
- DNS server hashes „**`b`**“ → **`c123..aad3`**
- DNS server searches and sends the suitable NSEC3 RR (incl. signature):
  `a25c..a018` `86400 IN NSEC3` **`177d..7f7e`** `( … )`

- Attacker gathers information: „After a host with the hashed name **`a25c..a018`** there is another host with the hashed name **`177d..7f7e`**“

→ As the hash function is a one way function, the attacker can not easily map the hashed values back to a domain name.

Summary

- DNS is one of the most important services deployed in the Internet
  - Mapping Name → IP
  - Distributed Name Database
  - Extensible
- The security of DNS is highly relevant for the security of the Internet
- DNSSEC is used for adding the missing security to DNS

Part II: Security of Components

- Part II: Security of Components

Major Problem: The Security of Components

- As already said: Users are often inexperienced!
  - Their skills in maintaining networks / computers, especially regarding security is low.
- Therefore we need special security components that can work autonomously even in *insecure* environments
- Some Requirements:
  - Integrity of infrastructural components, e.g. of the home CA
  - Protection of keying material (of the network / of the users), e.g. the home key
    - Prevents identity theft, abuse of services, ...
- Idea: Again use enterprise-grade security mechanisms:
  - Trusted Computing Technology

Background: TPM

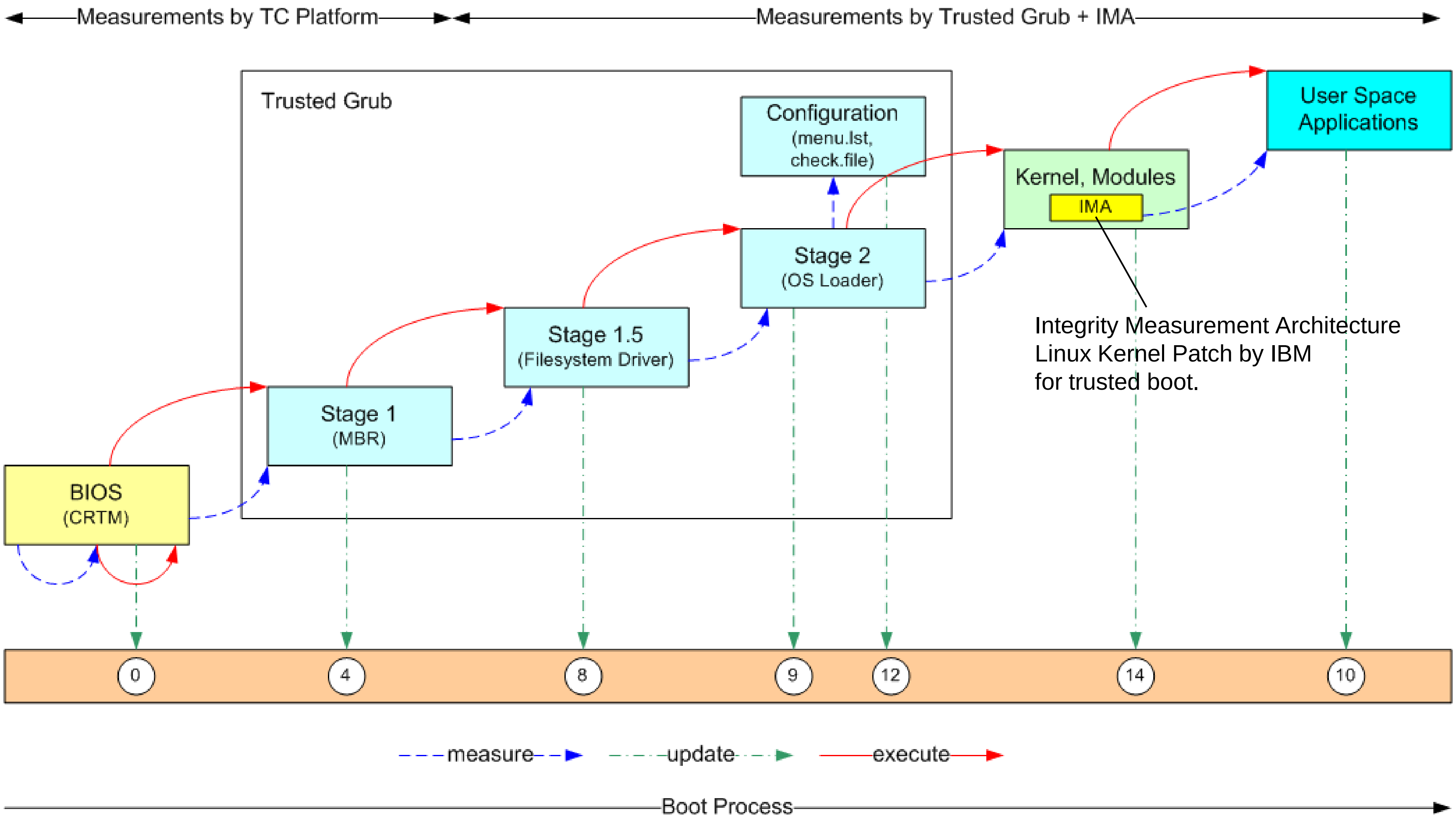
- Trusted Platform Module:
  - A cryptographic chip attached to mainboards
  - Specified and standardized by Trusted Computing Group (TCG)
- Provides
  - Unique system identity
  - Secure RSA key management (key creation, signing in shielded hardware)
  - Tamper proof key storage
  - Tamper proof memory for platform integrity measurements
- Access Restriction
  - Multiple Passwords, e.g. for owner authentication
  - Limitation / Danger: Malware may be able to intercept password and use TPM functionality for its own purpose.

TPM

- TPM is a solid foundation for many security mechanisms, for instance:
  - Secure Boot
    - The system will boot only, if the basic system components have integrity
    - Protects against pre-OS attacks, e.g. malware resident in MBR, Boot Loader, etc
  - Remote Attestation
    - Related to the Secure Boot
    - Idea is to proof to another device that the integrity of the software running on the computer is ok
    - Concept is based on a chain of fingerprints that represent the binaries on disk of the processes that were started
      - Only system and programs in a certain version are accepted.
  - Safeguard for keying material, e.g. the homes CA
    - Private Key does not leave TPM (normally)

Trusted Boot: Measuring System Components

- CRTM = Core Root of Trust for Measurement
  - Trust is bootstrapped from this anchor point, first measuring, then starting the processes



Enforcing boot-time integrity: Secure Boot

- Binding paradigm:
  - TPM can generate special RSA keys („Binding Keys“)
  - The private part of this key can only be used in a specific, trustworthy system configuration
  - The public part of this key is used to encrypt security relevant data

- Approach:
  - Encrypt the system using the public part of the Bind Key: inital RAM disk, Kernel, file system, ...
  - When system has booted up to Stage 2 of the Boot Loader, the remaining system can be started only if the private part of the Bind key can be used
    - = when the basic system components have integrity
  - ➔ **This will prevent that a compromized system boots**

- Other Secure Boot mechanisms exist: e.g. based on Intel TXT or UEFI technology

Even more problems, ...

- Secure Boot concepts only protect the boot time
- When the system is up and running the integrity of the system can be violated by many attacks. Just to mention some attack families:
  - Buffer Overflow Attacks
  - Attacks on the control flow of applications (Return-to-libc, ROP, ...)
  - Hijacking external function calls („GOT-Hijacking“)
  - Modification of code in memory (e.g. via debugger)

- Typically protection mechanisms only cover one specific attack
- Often they can be circumvented by new attacks

- Host Intrusion Detection Systems / Host Intrusion Prevention Systems try to detect / prevent infections
  - Unfortunately they run on the productive systems and can be attacked as well

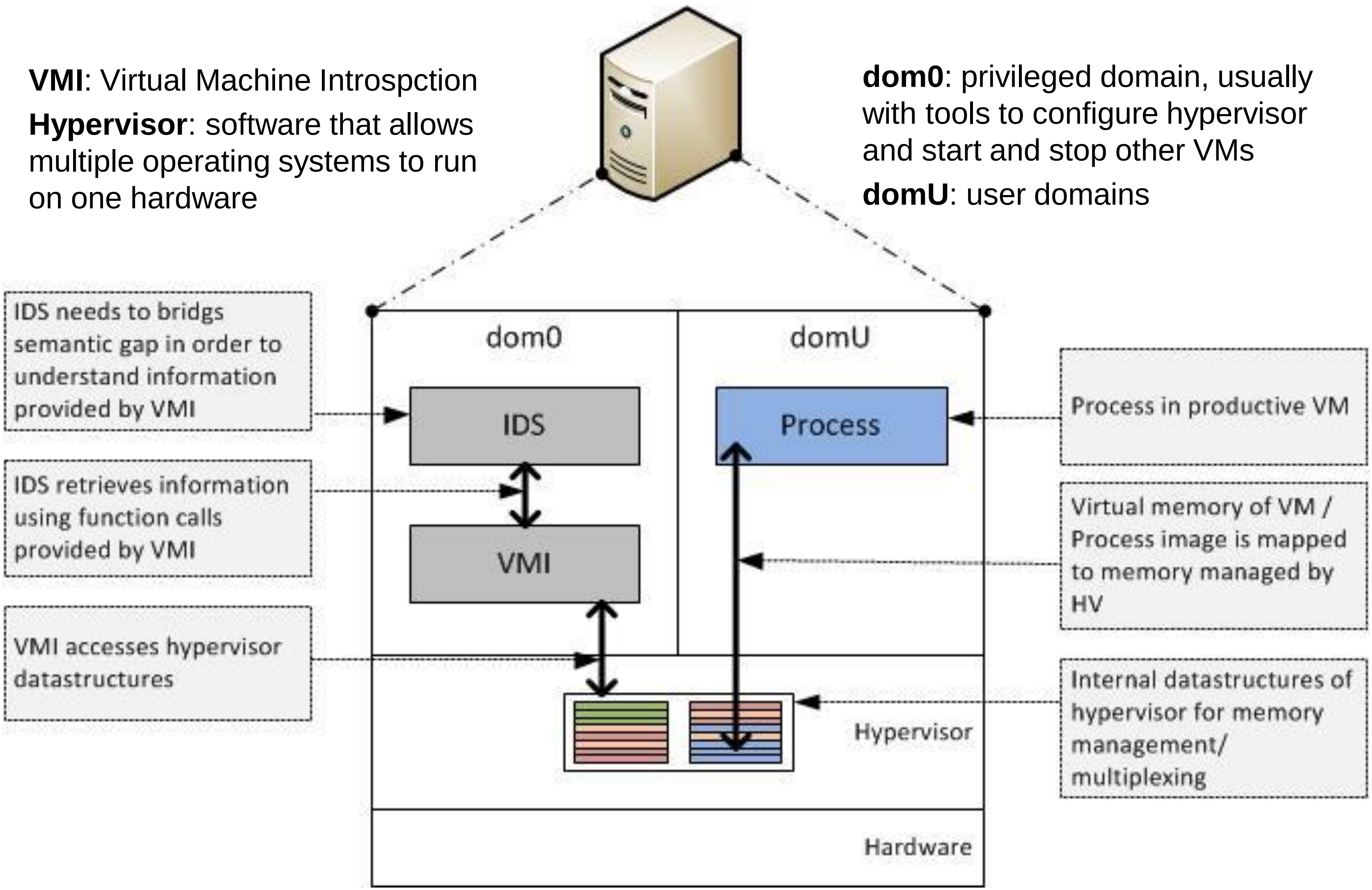
Virtual Machine-based Intrusion Detection

VMI: Virtual Machine Introspction

Hypervisor: software that allows multiple operating systems to run on one hardware

dom0: privileged domain, usually with tools to configure hypervisor and start and stop other VMs

domU: user domains



Virtual Machine-based Intrusion Detection

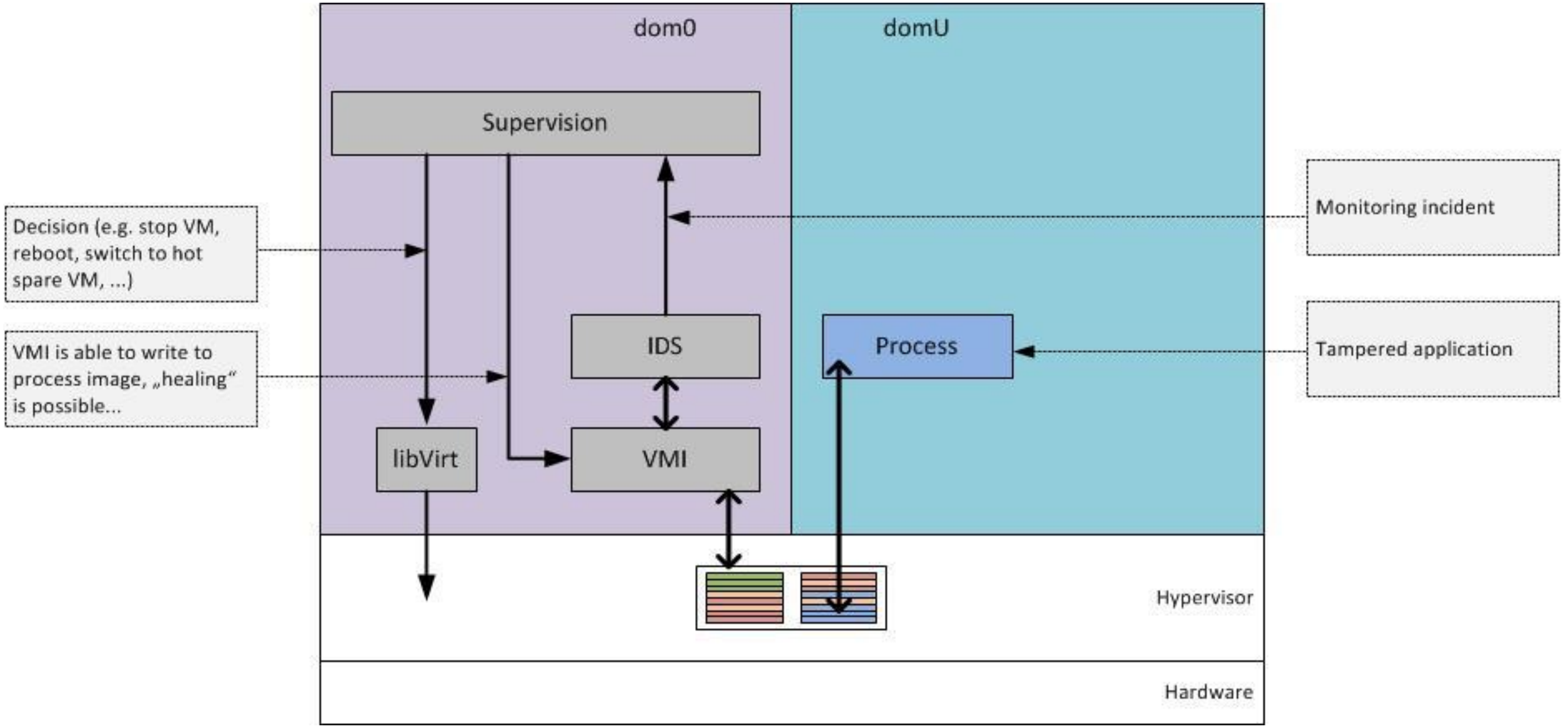
- VMI-based IDS run in isolated environments
  - VMI = Virtual Machine Introspection
    - Access memory of other VMs with the help of the hypervisor
  - Benefit: productive system and monitoring system are separated using a hypervisor

- The IDS is able to detect attacks.
- But we also need to be able to react in a proper way

- Trigger counter measures → idea: Use autonomous control loop („MAPE“-Cycle)
  - Measure: Obtain RAW data from monitored system
  - Analyze: Analyze the data, detect attacks
  - Plan: Decide which action can be applied
  - Execute: Execute the counter action

Triggering Counter Measures

- IDS analyzes raw data and is able to detect attacks.
- Attacks are sent to Supervision entity which decides how to react
- On example would be to disable network access of the attacked domain, e.g. using libVirt and „heal“ to infected process image via the VMI-tool



Outlook

- We‘re launching several projects related to the topics / discussions presented here in close future. Focus:
  - Virtualization
  - Attack detection
  - Counter measurements on attacks

- We‘re offering BA/MA theses and HiWi jobs to questions related to
  - Virtualization
  - Intrusion Detection / Prevention Systems
    - Host-based
    - VMI-based
  - ...

- If you are interested contact kinkelin / niedermayer @net.in.tum.de

Part III: Anonymity

- Part III: A brief introduction to Anonymity

Motivation

- Alice and Bob communicate using encryption.

  → Eve cannot read the data Alice and Bob are sending.

  But…

  → Eve knows that Alice and Bob are communicating.

  → Eve knows the amount of data Alice and Bob are sending. Alice observes the traffic patterns.

  - e.g. Bob as Webserver may sent the page which is fingerprinted in having 13kB of data, and 13 included objects with size from 2kB to 117kB.
    → Eve knows what Bob is sending to Alice
    → encryption not sufficient for static content

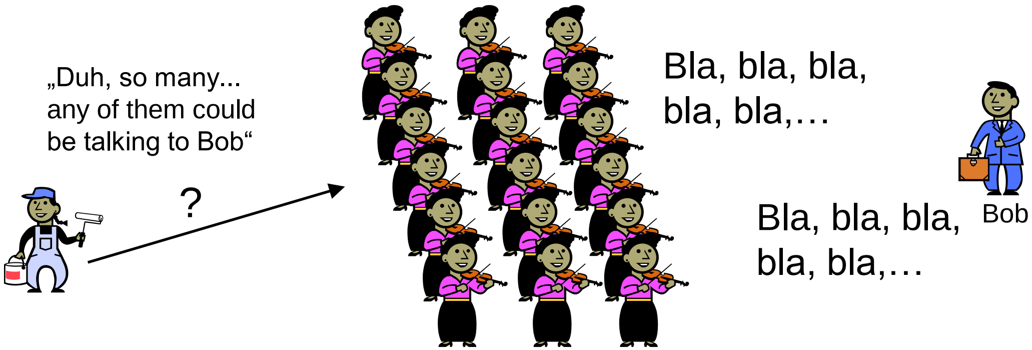
Anonymity

„Anonymity is the state of being not identifiable within a set of subjects, the anonymity set.“

Andreas Pfitzmann et. al.

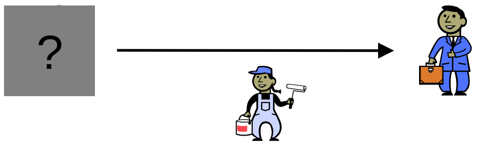
Anonymity Set

- The set of all possible suspects who might cause an action.
- The larger the anonymity set, the better the anonymity.
  - ... not completely true. Also, the more equal the probability for the suspects in the set, the better.



Terminology

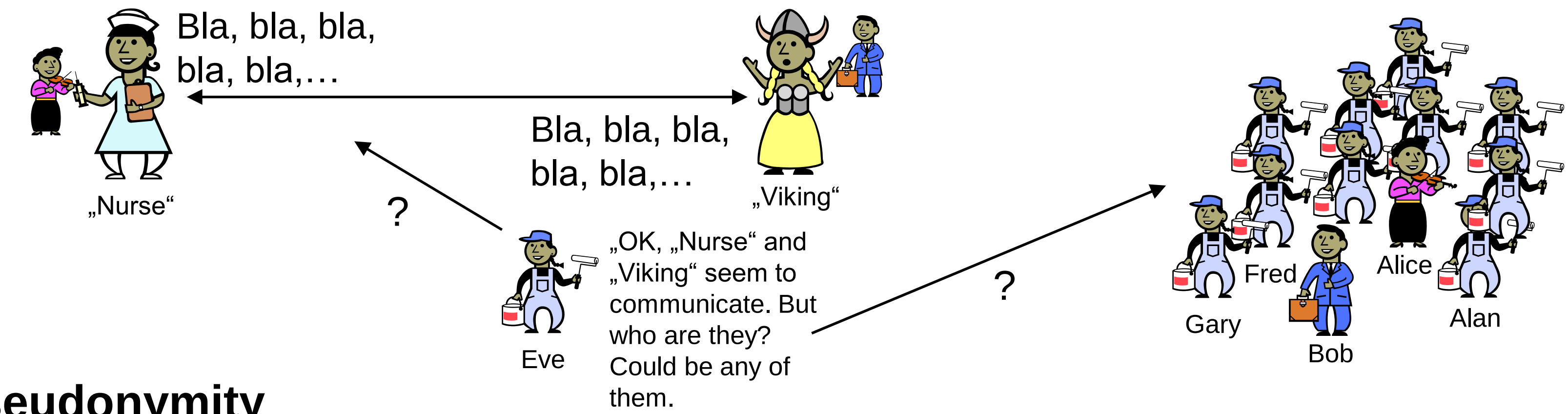
- Sender Anonymity
  - The initiator of a message is anonymous. There may be a path back to the initiator.
  - „*???* to Bob“



- Receiver Anonymity
  - The receiver of a message is anonymous.
  - „Alice to *???*“

- Unlinkability
  - The observer cannot decide who is communicating with whom.
  - „*???* communicates with *???*“

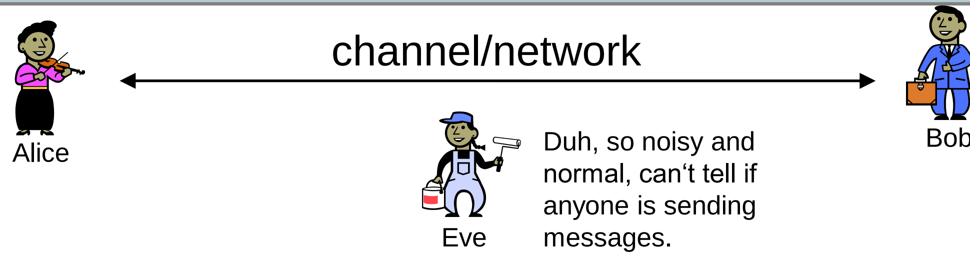
Pseudonymity



Pseudonymity

- A pseudonym is an identity for an entity in the system. It is a „false identity“ (word origin of pseudonym) and not the true identity of the holder of the pseudonym. The holder hides the true identity behind the pseudonym.
  - e.g. a nickname in a forum, random string in an anonymity system
- Noone, but a trusted party may be able to link a pseudonym to the true identity of the holder of the pseudonym.
- A pseudonym can be tracked. We can observe its behaviour, but we do not know who it is.
  - „Nurse“ is always „Nurse“.
  - vs. anonymity: In anonymous systems, we cannot say if it is the same user „Nurse“ again. An anonmyous entity is indistinguishable from all other anonymous entities.

Unobservability / Covert Channel



Unobservability

- „Unobservability is the state of items of interest being indistinguishable from any item of interest at all. “ (according to Andreas Pfitzmann et. al)
- Eve will not see a different channel behaviour if Alice and Bob communicate or not.

Covert Channel

- An observer cannot tell from observing the network if there is communication or not.
- A covert channel is hidden within the noise of a system or in legitimate normal communication and its normal patterns.
- Methods
  - Spread Spetrum Methods in Noisy Channels
  - Steganography
  - Hide in normal (preferably encrypted) communication.
  - …
- Discussion
  - Either extremely slow or statistical patterns uncover the channel.
  - Connecting to an anonymous system and hiding traffic patterns is not a covert channel.
  - A normal HTTP/HTTPS connection from Alice to Bob is also not a covert channel.

Adversary Models

Basic adversary characteristics

- Position
  - External: „sits“ on the wire
  - Internal: participates in the anonymous system
- Geographic
  - Global: sits on all wires
  - Local: sits on some local wires
  - Partial: controls parts of the network
- Participation
  - Passive: only observes traffic
  - Active: may send, modify, and drop messages.

Adversary Models

Typical adversary models

- Global Passive Adversary (GPA)
  - Observes and efficiently analyses the complete network.
  - No active participation in the network.
  - External attacker.
- Global Active Adversary (GAA)
  - Also performs active attacks.
- Partial Passive Adversary (PPA)
  - Observes only parts (<< 50 %) of the network.
  - External attacker.
- PPA or GPA with some active nodes
  - Add some internal nodes that may also perform active attacks.
- Local observer
  - An observer that locally observes the endpoints of a communication.

→ *All of these attacker models are too strong for current realtime low-latency anonymous networks.*

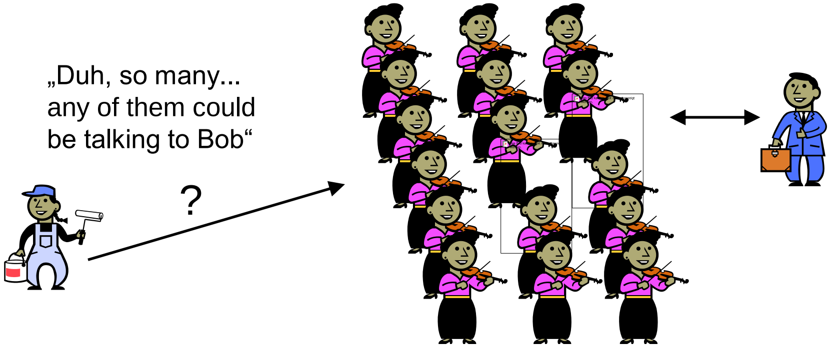
Measuring Anonymity

How anonymous is a systems?

- Number of known attacks?
- Lowest Complexity of successful attacks?
- Information leaked though messages and maintenance procedures?

Examples

- Anonymity set
  - Anonymity Set = |{suspects}|
  - Suspects are all entities that could have sent / received / participated.
  - In the example, the anonymity set is 18.
  - Limitations
    - No way to include meta knowledge.
      - An attacker could know that Alice is more likely to communicate with Bob than others because she is an attacker in a security lecture ;).

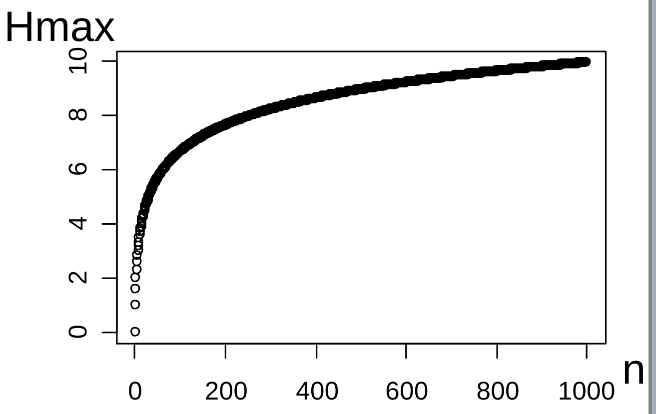


Measuring Anonymity

So, we are an attacker in a security lecture. For talking with Bob, we use this knowledge to conclude Alice 0.9 and other 100 suspect 0.001.

Any metric for that?

- Entropy
  - Combines the number of suspects and their probabilities in one metric.
  - Let p_i be the probability for suspect i.
  - Entropy $$H = -\sum_i p_i ld(p_i)$$
  - Entropy is maximized for a fixed number of suspects if all are equally likely (p_i=1/n for all i) → Hmax=ld(n)
  - e.g. 101 nodes as above Hmax = 6.7, if we use meta knowledge with probability p_alice=0.9 then H=1.1.

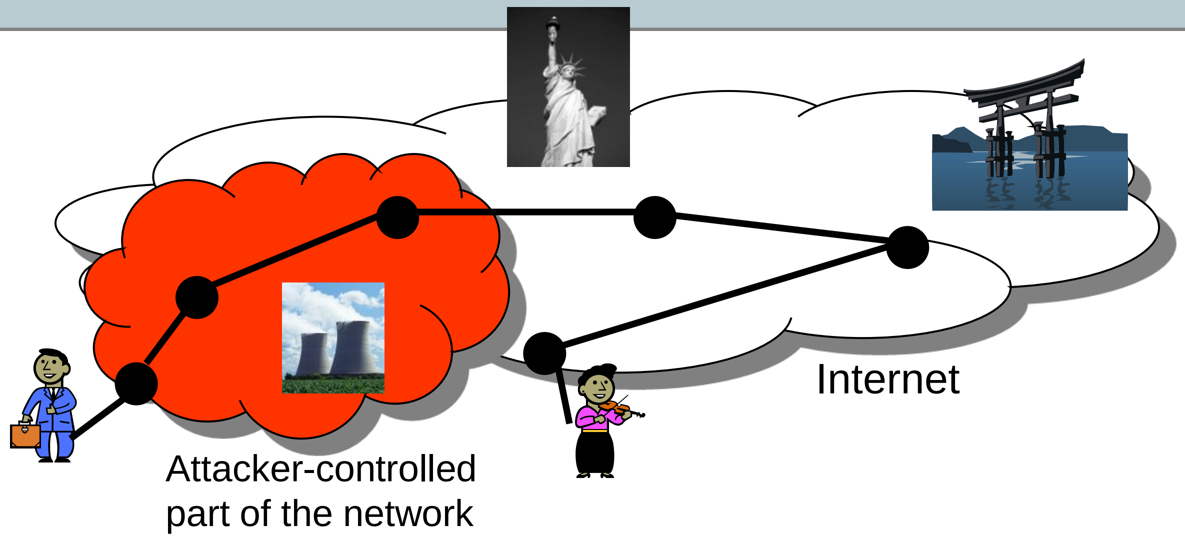


Basic concepts for anonymous systems

Basic concepts for anonymous systems

- Escape geographically (→ Re-Routing)
- Confuse packet flows at re-routers (→ Mixing)
- Hide content (→ Layered Encryption and Hop-by-Hop encryption)
- Hide message properties (→ Padding)
- Hide communication / flow properties (→ Dummy Traffic)

Re-Routing



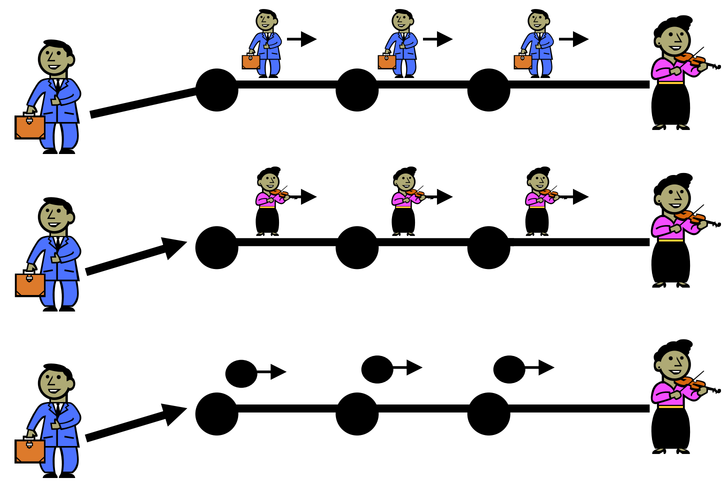
Re-Routing

- Anonymity requires to hide sender/receiver relationships. As a direct message would be such a relationship, anonymity requires to route message via other intermediate nodes (*re-routers*).
- With respect to fighting an attacker, re-routing tries to get the message out of the area controlled by the attacker. The idea is to globally espace a partial attacker („*escape geographically*").
- Messages need to be encrypted.
  - Otherwhise, attacker can simply read source/target locator.
  - Usually, re-encryption hop-by-hop. → Packet looks different on each path section.

Path Selection Strategies

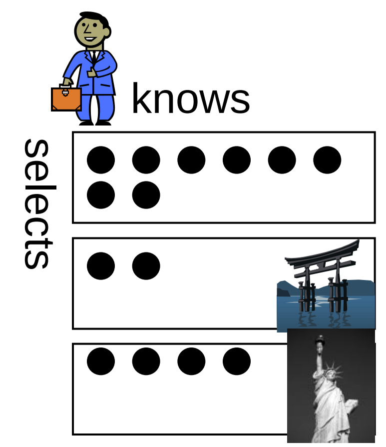
Who selects?

- Sender
  - The sender initiates a path hop-by-hop.
    → „Sender controls her anonymity"
- Receiver
  - The receiver initiates a path from some rendezvous point to herself hop-by-hop.
    → „Receiver controls her anonymity"
- Re-router
  - Each re-router selects the next hop for a path.
  - Problem: An internal attacker may select other attackers.
- Network design
  - The route is fixed by the system itself.



Selection

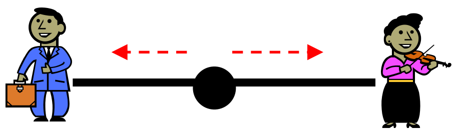
- Selection requires knowledge of large set of re-routers.
- Random selection provides most entropy.
- Biased selection strategies
  - Geographic diversity of used re-routers (→ Optimize trust, escape attacker geographically).
  - Organizational diversity of used re-routers (→ Optimize trust).



1 Hop (simply proxy)

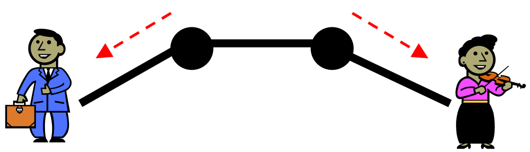
- Trust problem as proxy knows everything.
- Trusted proxy may leak meta-information about those who trust it.

e.g. trust-proxy-tuebingen may imply
„someone in Tübingen“ … hmm…
only Bob is from Tübingen → Bob



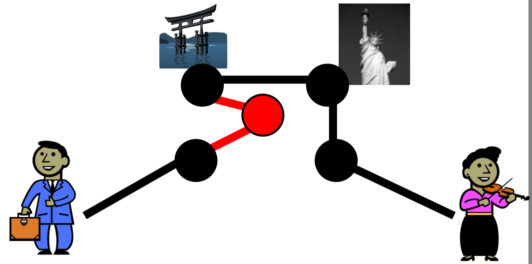
2 Hops

- No hop knows sender and receiver.
- But each hop likely to know its position on path.



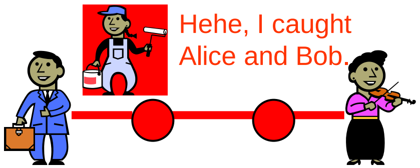
More hops

- Position on path for a re-router less clear.
- Better diversity / but more likely to select attacker.



Fixed length vs. random length

- Random length makes attacks based on positions in the path harder.



Fixed with length 2 → attack

Random length → could also be

Re-routing

Other aspects

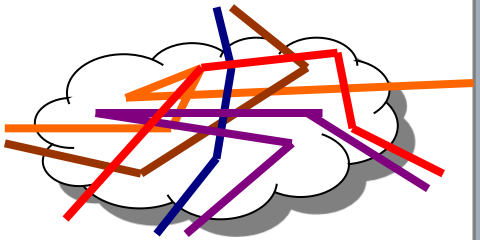
- Degree of freedom for path selection (Topology)
  - A high degree has advantages with respect to trust.
  - A low degree better hides communication properties as many flows follow identical paths.
- Lifetime of a path – fixed path vs dynamic path
  - Fixed path
    - Use same path for entire session.

    \+ performance, overhead, no need to change good path

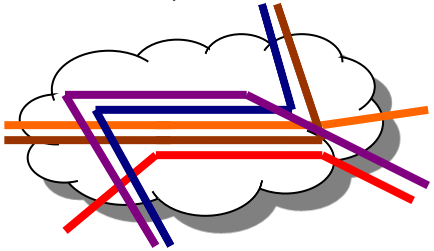
    \- easier to observe for an attacker
  - Dynamic path
    - Change path frequently during session.

    \+ makes (long-term) observations harder

    \- with internal attackers, the more often a path is changed the more likely it is to hit a path solely consisting of attackers.

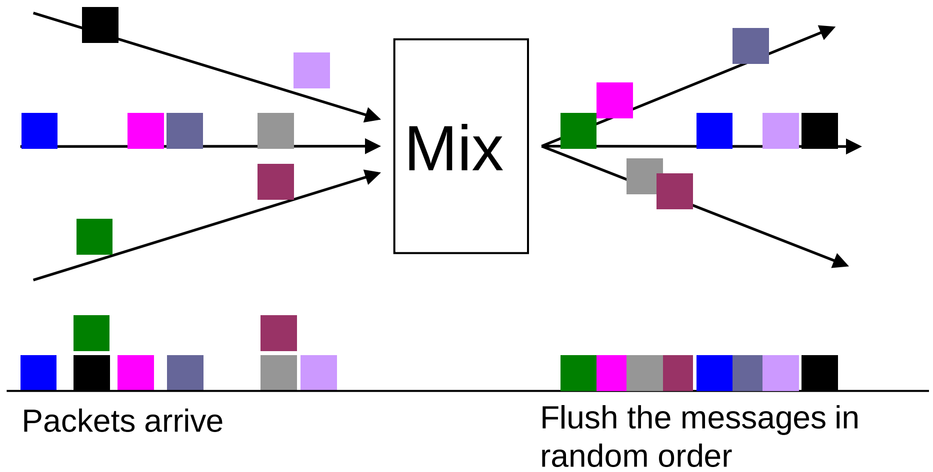


Completely free
→ Chaotic but on often only one flow per section



Strongly path resistricted
→ More overlaps of flows

How does a re-router operate?

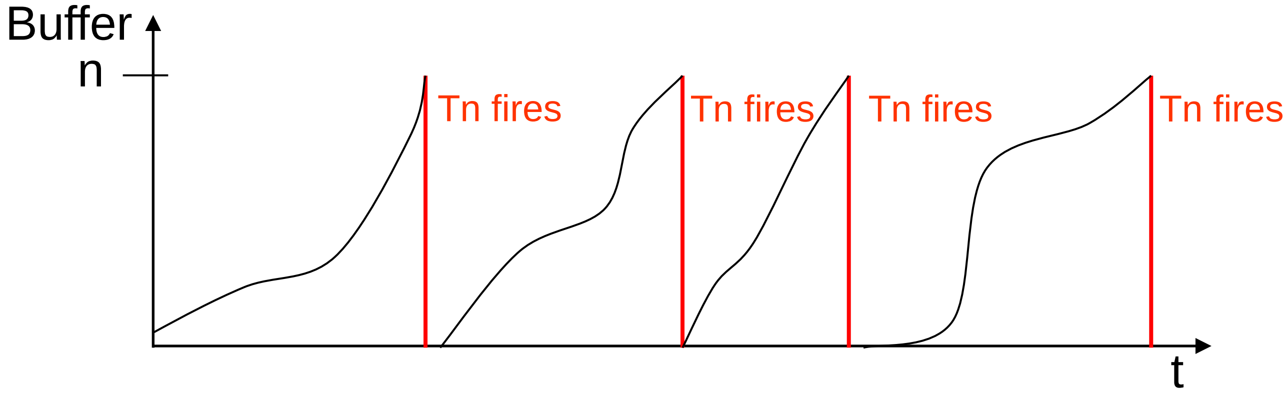


Assumption

- Packets change appearance -> re-encryption

Mix

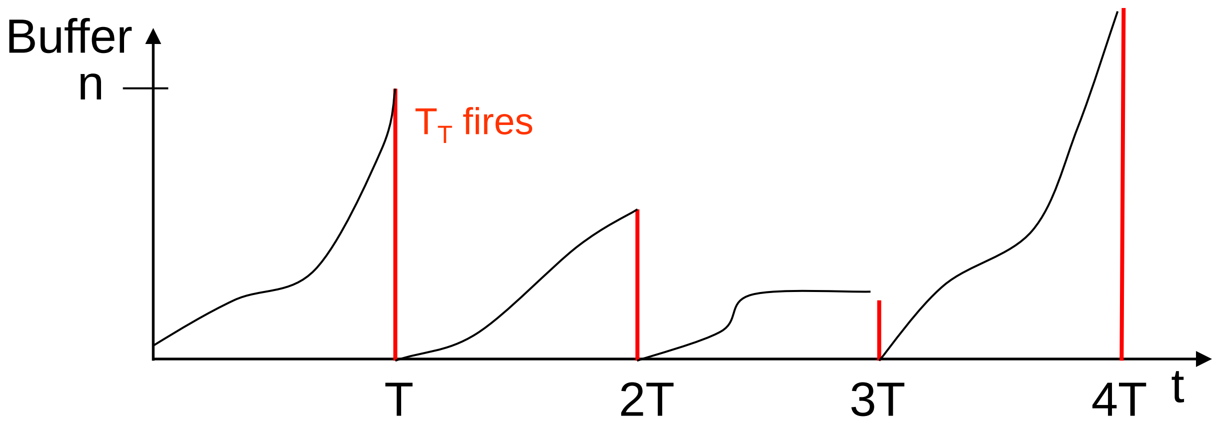
- Concept by David Chaum (1981)
- A mix is a re-router that does not directly forward messages. A mix first collects a number of messages and then sends them out in random order.
- An attacker observing a mix cannot tell which incoming messages is which outgoing message („escape through re-ordering“).



Threshold Mix

- A threshold mix T_n with threshold n.
- Operation
  - T_n collects messages until it buffers n messages.
  - Then it fires = T_n sends these n messages in random order.
- Anonymity Set = n.
- Performance depends on rate of incoming messages.

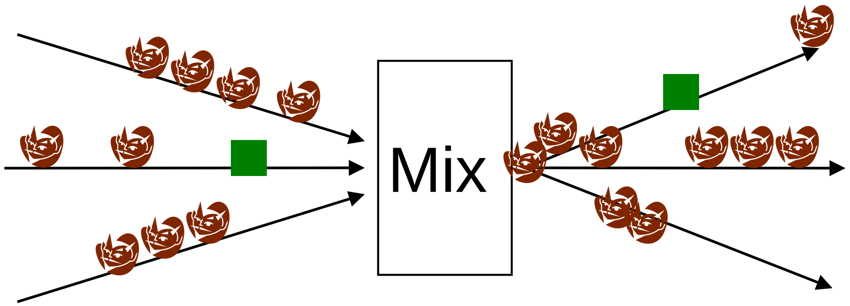
Timed Mix



Timed Mix

- A timed mix T_T with interval time T.
- Operation
  - T_T collects messages for time T.
  - Then it fires = T_T sends these messages in random order.
- Anonymity Set = number of messages that arrived in interval
  - Can be small (1 = no anonymity) or large („buffer capacity of mix“). → Anonymity depends on rate of incoming messages

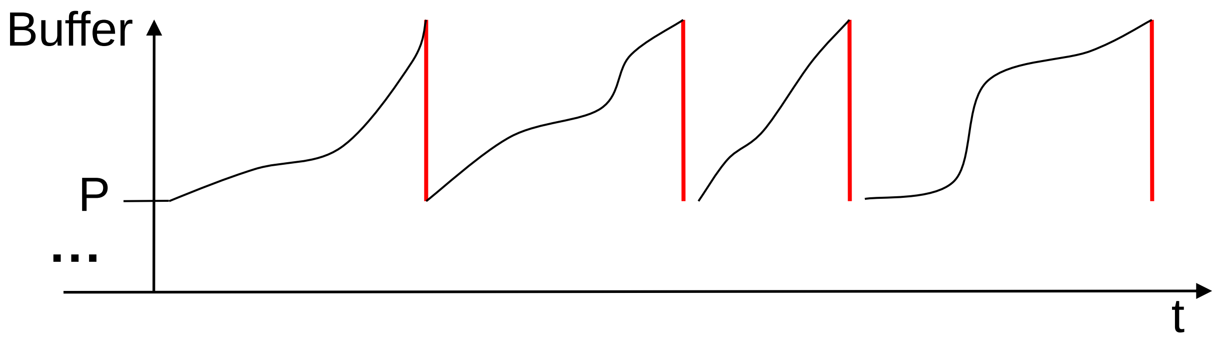
n-1 attack on mixes



n-1 attack on a mix

- An n-1 attack is an active attack.
- Basic idea
  - The attacker inserts messages and degrades the anonymity set.
- Attack situation
  - n messages arrived at mix
  - n-1 messages are from the attacker
- The mix fires.
  - Attacker knows its n-1 messages, can identify the other one.
- Basic form is against threshold mix, but a strong attacker could also delay messages towards a timed mix.

Pool Mix / Exponential Mix



Pool Mix

- Basic idea
  - To increase anonymity set and to make the n-1 attack more difficult, ensure that always a pool of P old messages is in the mix.
- Operation
  - Collect messages and fire at some point in time (threshold/timed/…).
  - With S messages in the buffer, randomly select S-P and send them in random order.

Exponential Mix

- Mix messages by randomly-delaying. No firing.
- Operation
  - Message Mt arrives at time t.
  - Add a random delay D (exponential distribution / geometric distribution) and schedule message for time t+D.
  - Send Mt at scheduled time t+D.

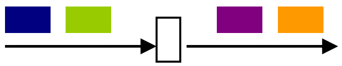
Discussion of Mixes

Discussion

- *When a message passes a set of mixes, one honest mix is enough to provide anonymity! (for the message)*
- Mixes protect single messages.
  - Flows with several messages may be identified due to their traffic volume.
- To ensure performance or a good anonymity set, a mix needs a lot of traffic.
  - Not suitable for decentralized approaches that opt for low-latency.
- The operation of a mix is targeted against a strong observer that controls all interfaces of a mix or all mixes in a mix network.
  - Maybe an overkill for overcoming realistic attackers in combination with the use of re-routing.
  - Most low-latency anonymity systems only re-route and do not mix.
- Re-routers with lots of traffic also slightly randomize order due to internal processing and queuing (despite FIFO and Round Robin).

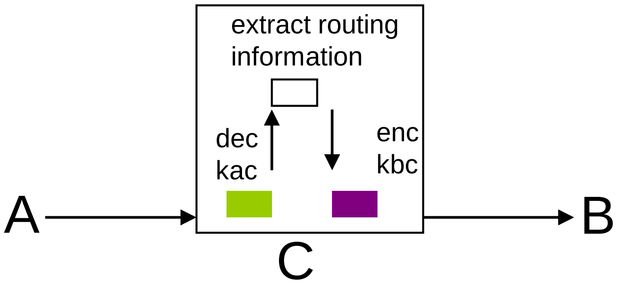
Goals

- Hide the content from observers.
- The outgoing message from a re-router should look different than the corresponding incoming message.



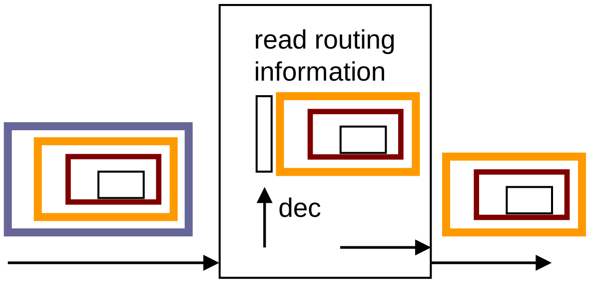
Hop-by-Hop encryption

- Each hop decrypts (key with predecessor) and re-encrypts (key with successor) message.
- End-to-end message confidentiality can be achieved by adding end-to-end encryption.
- Discussion
  - Re-routers see identical packets → internal attacker
  - Difficult to implement unless re-routers select paths.

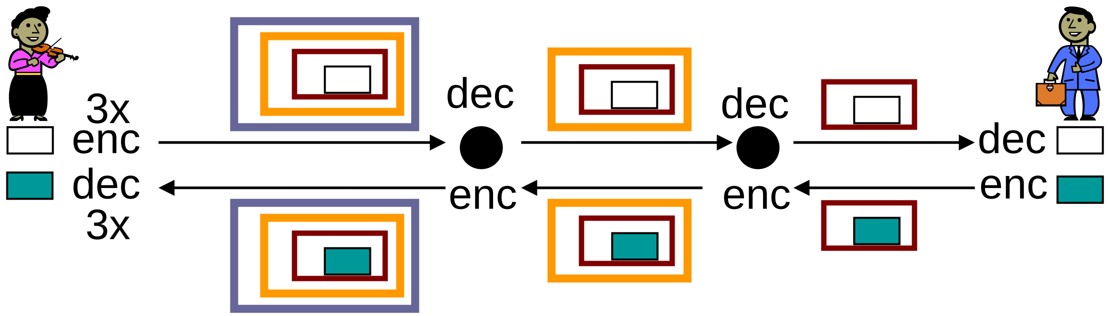


Layered encryption

- Sender encrypts message several times with keys for all hops. It adds a layer of encryption over the message for each hop.
- Either public key of re-router or an established shared key between sender and re-router.
- Re-routers decrypt the message to determine next hop and send the decrypted message.



Onion Routing



Onion Routing

- Onion Routing is based on layered-encryption.
- The term is a metaphor for the operation of such routers as the packets is peeled like an onion.
- Onion routers (ORs) do not mix or delay packets. They usually operate with simple FIFO or round robin (between flows) queues.
- Pad message to constant length at each hop.

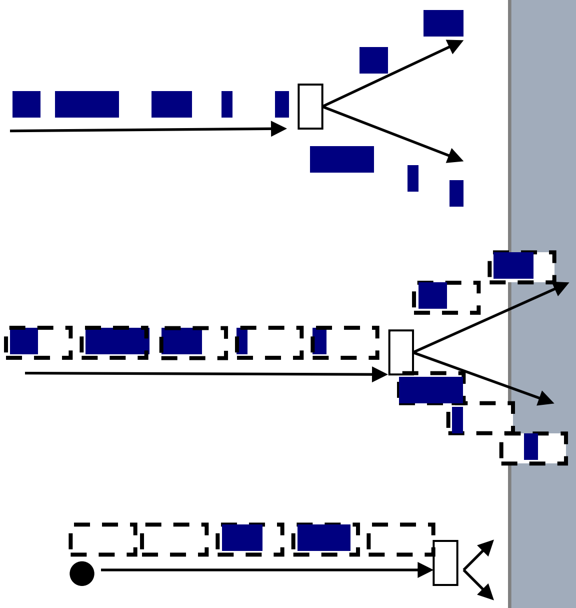
Keys

- Public keys of re-routers (not very efficient).
- Sender/Initiator uses public key of re-routers for path establishment and establish shared key with each re-router on the path.

Padding / Dummy Traffic

Padding

- Message size
  - can be used to fingerprint messages.
  - unveils information like positions in a path
- Message Padding
  - Add padding (random data) to smaller packets so that all packets are of identical size.
  - → Necessary and thus widely used in anonymity systems
- Link Padding
  - Use dummy messages to pad the link to a constant bandwidth.
  - → Necessary against global and local observers, used in some systems. Link padding is covering the existence of real traffic.



Dummy Traffic

- Send dummy traffic through the network to hide traffic volumes of flows and cover real traffic.
  - Link padding is a subclass of dummy traffic.
- Except for link padding, dummy traffic is hardly used in anonymity systems → usually considered too expensive for too little gain.