Decentralized Public Key Infrastructures

Christian Grothoff

Berner Fachhochschule

24.5.2024

Learning Objectives

Part I: Introduction to GnuPG

PGP

- PGP can be used to encrypt and digitally sign files and e-mails.
- Data is at rest or transmitted unidirectionally. $\Rightarrow$ No secure channel!
- PGP was published by Philip R. Zimmermann[1] in the early 1990s.
- Got immediate NSA attention and encountered legal issues on its use of RSA cryptography patents.
- PGP certificates are public key certificates with one or more identity labels tied to it.

[1] `http://www.philzimmermann.com/`

GnuPG

- Free version of PGP, with library (libgcrypt)
- Provides common cryptographic primitives
- Provides implementation of OpenPGP ([4, 5, 3])
- Commonly used for:
  - secure E-mail (authentication, encryption)
  - encrypt files
  - sign files — i.e. sources and binaries in Free Software distributions

PGP Certificate Overview

PGP Version identifies which version of PGP was used to create the key associated with the certificate

Holder's public key the public portion of your key pair, together with the algorithm of the key: RSA or DSA (Digital Signature Algorithm)

Holder information this consists of "identity" information about the user, such as their name, user ID, photograph, and so on...

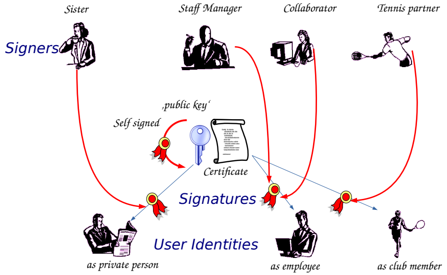
Holder digital signature also called a self-signature, this is the signature using the corresponding private key of the public key associated with the certificate

Validity period the certificate's start date/ time and expiration date/ time; indicates when the certificate will expire

Preferred symmetric encryption algorithm indicates the encryption algorithm to which the certificate owner prefers to have information encrypted. The supported algorithms are CAST, IDEA, Triple-DES, AES, ...

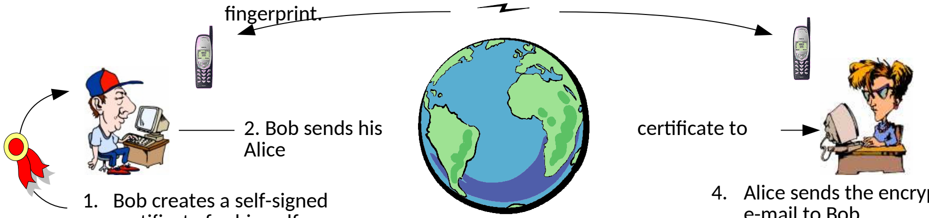
PGP Certification

- One certificate may be signed by multiple entities (persons).
- Digital signatures may bind different user attributes to a certificate when verifying the authenticity of that user.



People apply different methods to check authenticity before signing a key!

PGP Key Signing



3. Alice verifies fingerprint.

2. Bob sends his Alice

certificate to

1. Bob creates a self-signed certificate for himself.

4. Alice sends the encrypted e-mail to Bob.

Trust on First Use (TOFU)

Another kind of direct trust security model:

- Client creates a trust relationship with a not-yet-trusted and unknown endpoint.
- The public key of the endpoint is not verified, but *subsequent* connections to the same peer require the public key paired with other information of the service to remain the same.

TOFU is typically used in SSH and in HTTP Public Key Pinning (HPKP).

The Web of Trust

Problem:

- Alice has certified many of her contacts and *flagged* some as *trusted* to check keys well.
- Bob has been certified by many of his contacts.
- Alice has **not** yet certified Bob, but wants to securely communicate with him.

[2] Simplified, details later.

The Web of Trust

Problem:

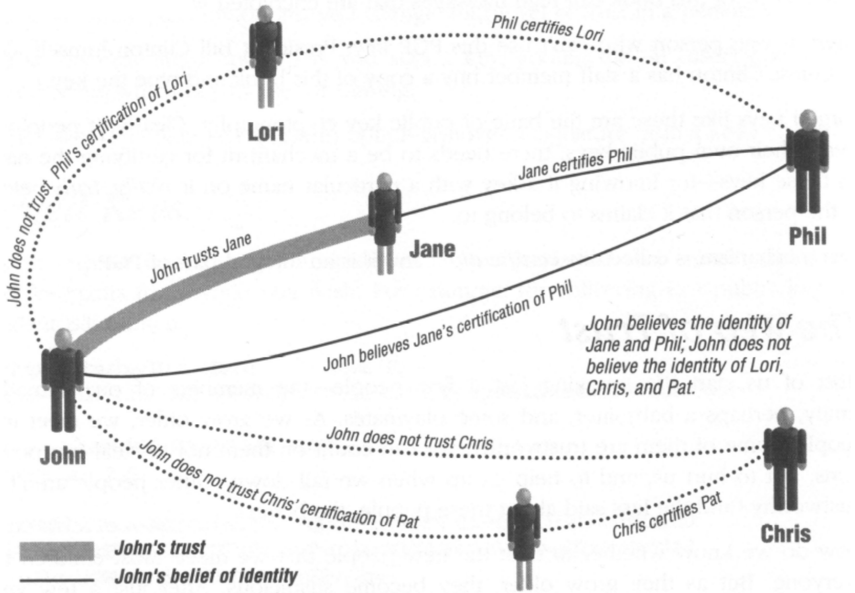
- Alice has certified many of her contacts and *flagged* some as *trusted* to check keys well.
- Bob has been certified by many of his contacts.
- Alice has **not** yet certified Bob, but wants to securely communicate with him.

Solution:

- Find paths in the certification graph from Alice to Bob.
- If sufficient number of short paths exist certifying the same key, trust it.[2]

[2] Simplified, details later.

The Web of Trust


Phil certifies Lori
Lori
John does not trust Phil's certification of Lori
Jane certifies Phil
Phil
Jane
John trusts Jane
John believes Jane's certification of Phil
John believes the identity of Jane and Phil; John does not believe the identity of Lori, Chris, and Pat.
John
John does not trust Chris
John does not trust Chris' certification of Pat
Chris certifies Pat
Chris
John's trust
John's belief of identity

The PGP Private Keyring

Stores private/public key pairs:

- timestamp
- key ID (indexed)
- public key
- encrypted private key (with passphrase)
- user ID (indexed)

The PGP Public Keyring

Stores public key pairs, certificate and trust status:

- timestamp
- key ID (indexed)
- public key
- user ID (indexed)
- owner trust:
  - unknown user
  - usually not trusted to sign
  - usually trusted to sign
  - always trusted to sign
  - ultimately trusted (own key, only present in private key ring)
- signature(s)
- signature trust(s); copy of owner trust of the signer
- validity of public key

Key validity calculation

- if at least one signature trust is ultimate, then the validity of the key is 1 (complete)
- otherwise, a weighted sum of the signature trust values is computed:
  - always trusted signatures has a weight of $1/x$
  - usually trusted signatures has a weight of $1/y$

x, y are user-configurable parameters, default $x = 1$, $y = 2$.

Certificate Trust Models (Summary)

Direct Trust	One trusts in a relationship between “public key” and “identity”, which it has verified by itself only. The identity of the subject (owner) is proven directly (personally).
Web of Trust	One accepts/applies “public keys”, where the identity binding is validated by others (persons or agents). One accepts other entities as trustworthy authorities (indirect trust or recommended trust).
Hierarchical Trust	One accepts/applies “public keys”, where the identity binding is validated by a trustworthy authority.

See also: **individualism**, **anarchism**, **authoritariansim**.

Certificate Trust Models (Summary)

Direct Trust
- Zero-solution: public key must be exchanged over 2nd/private channel or remain non-verifiable.
- Usable in limited scope. Key management is complex, legal validity/liability not possible.

Web of Trust
- Flexible solution: One applies public keys validated by other entities.
- Usable in bigger scope (e.g. community). Key management less complex using online key server. Legal validity/liability not possible.

Hierarchical Trust
- Strict solution: One applies public keys only if validated by a "trustworthy" authority.
- Usable in national or even global scope. Key management still complex but mostly done by experts. Legal validity/liability possible.

See also: **individualism**, **anarchism**, **authoritariansim**.

Using GnuPG

```
$ gpg --gen-key
$ gpg --export
$ gpg --import FILENAME
$ gpg --edit-key EMAIL
(gpg) fpr
(gpg) sign
(gpg) trust
$ gpg --clearsign FILENAME
```

Excercise: Explore

`https://pgp.mit.edu`

Part II: Distributed Hash Tables

Distributed Hash Tables (DHTs)

- Distributed **index**
- GET and PUT operations like a hash table
- JOIN and LEAVE operations (internal)
- Trade-off between JOIN/LEAVE and GET/PUT costs
- Typically use exact match on cryptographic hash for lookup
- Typically require overlay to establish particular connections

DHTs: Key Properties

To know a DHT, you must know (at least) its:

- routing table structure
- lookup procedure
- join operation process
- leave operation process

... including expected costs (complexity) for each of these operations.

A trivial DHTs: The Clique

- routing table: hash map of all peers
- lookup: forward to closest peer in routing table
- join: ask initial contact for routing table, copy table, introduce us to all other peers, migrate data we're closest to to us
- leave: send local data to remaining closest peer, disconnect from all peers to remove us from their routing tables

Complexity?

A trivial DHTs: The Circle

- routing table: left and right neighbour in cyclic identifier space
- lookup: forward to closest peer (left or right)
- join: lookup own peer identity to find join position, transfer data from neighbour for keys we are closer to
- leave: ask left and rigt neighbor connect directly, transfer data to respective neighbour

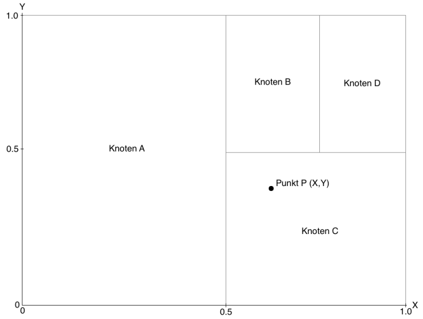
Complexity?

Additional Questions to ask

- Security against Eclipse attack?
- Survivability of DoS attack?
- Maintenance operation cost & required frequency?
- Latency? ($\neq$ number of hops!)
- Data persistence?

Content Addressable Network: CAN

- routing table: neighbours in *d*-dimensional torus space
- lookup: forward to closest peer
- join: lookup own peer identity to find join position, split quadrant (data areas) with existing peer
- leave: assign quadrant space to neighbour (s)

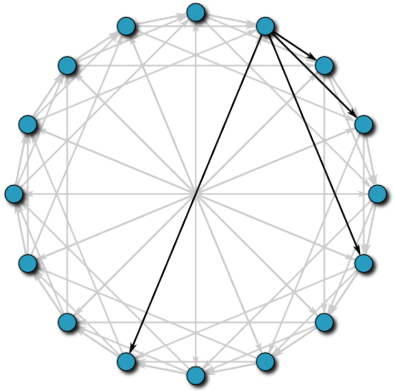


Interesting CAN properties

- CAN can do range queries along $\leq n$ dimensions
- CAN's peers have $2d$ connections (independent of network size)
- CAN routes in $O(d \sqrt[d]{n})$

Chord

- routing table: predecessor in circle and at distance 2^i, plus r successors
- lookup: forward to closest peer (peer ID after key ID)
- join: lookup own peer identity to find join position, use neighbor to establish finger table, migrate data from respective neighbour
- leave: join predecessor with successor, migrate data to respective neighbour, periodic stabilization protocol takes care of finger updates

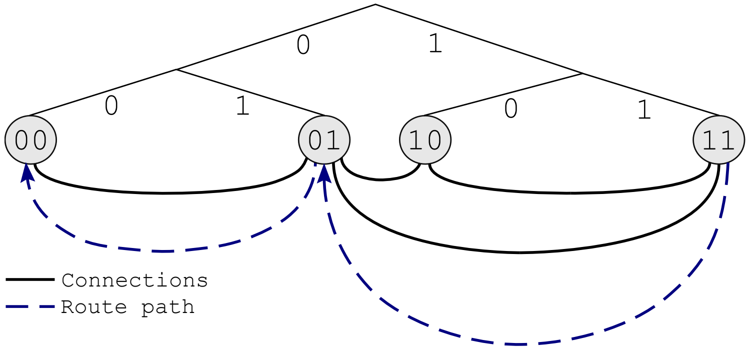


Interesting Chord properties

- Simple design
- $\log_2 n$ routing table size
- $\log_2 n$ lookup cost
- Asymmetric, inflexible routing tables

Kademlia

- routing table: 2^{160} buckets with k peers at XOR distance 2^i
- lookup: iteratively forward to α peers from the "best" bucket, selected by latency
- join: lookup own peer identity, populate table with peers from iteration
- maintenance: when interacting with a peer, add to bucket if not full; if bucket full, check if longest-not-seen peer is live first
- leave: just drop out

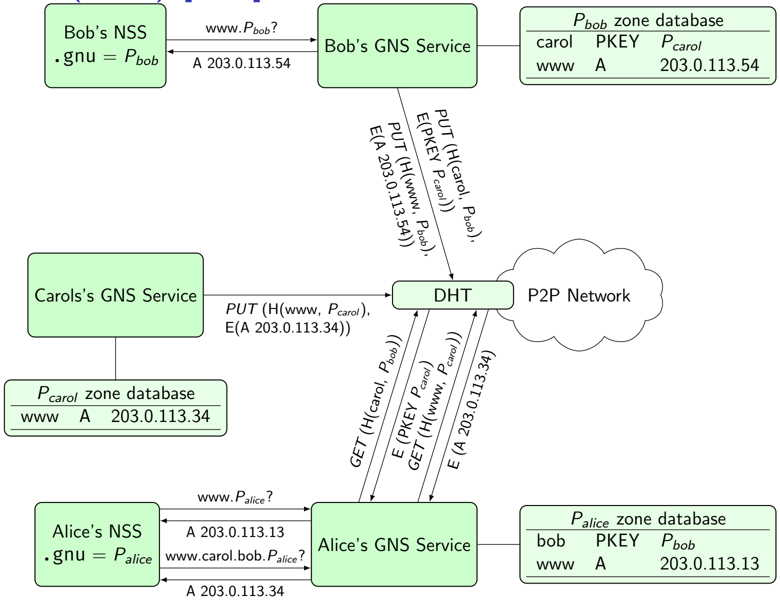


Interesting Kademlia properties

- XOR is a symmetric metric: connections are used in both directions
- α replication helps with malicious peers and churn
- Iterative lookup gives initiator much control,
- Lookup helps with routing table maintenance
- Bucket size trade-off between routing speed and table size
- Iterative lookup is a trade-off:
  - good UDP (no connect cost, initiator in control)
  - bad with TCP (very large number of connections)

Part III: The GNU Name System (RFC 9498)

The GNU Name System (GNS) [8, 6]



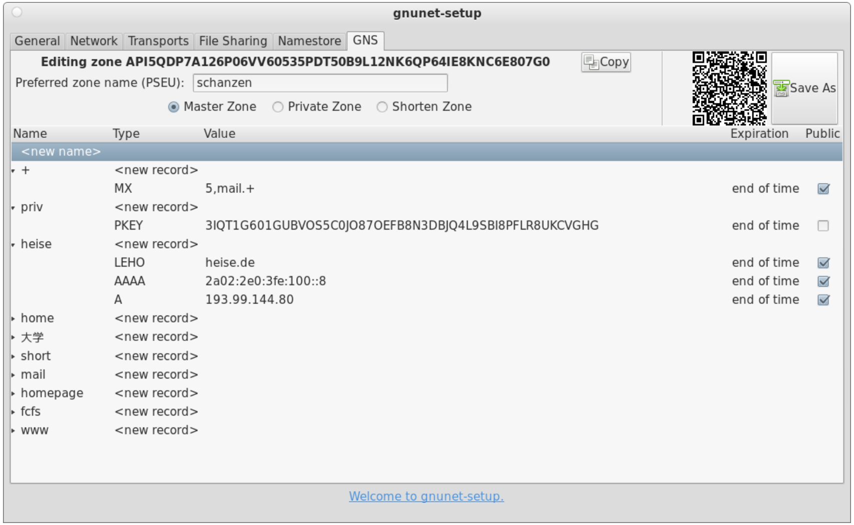
The GNU Name System[3]

Properties of GNS

- Decentralized name system with secure memorable names
- Delegation used to achieve transitivity
- Also supports globally unique, secure identifiers
- Achieves query and response privacy
- Provides alternative public key infrastructure
- Interoperable with DNS

[3] Joint work with Martin Schanzenbach, Matthias Wachs and Bernd Fix

Zone Management: like in DNS



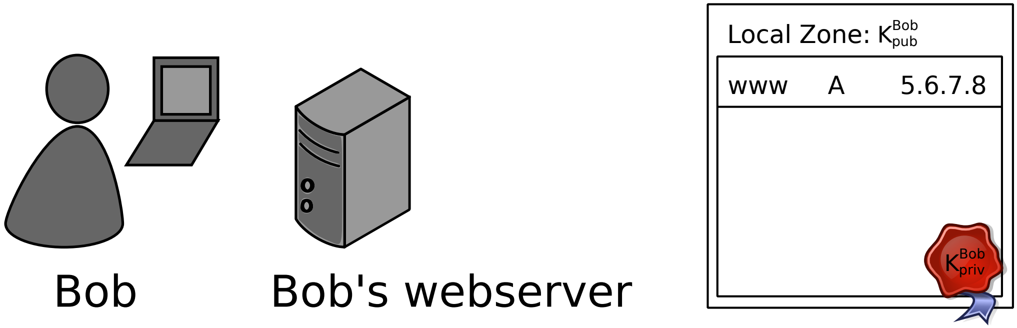
Editing zone API5QDP7A126P06VV60535PDT50B9L12NK6QP64IE8KNC6E807G0

Preferred zone name (PSEU): schanzen

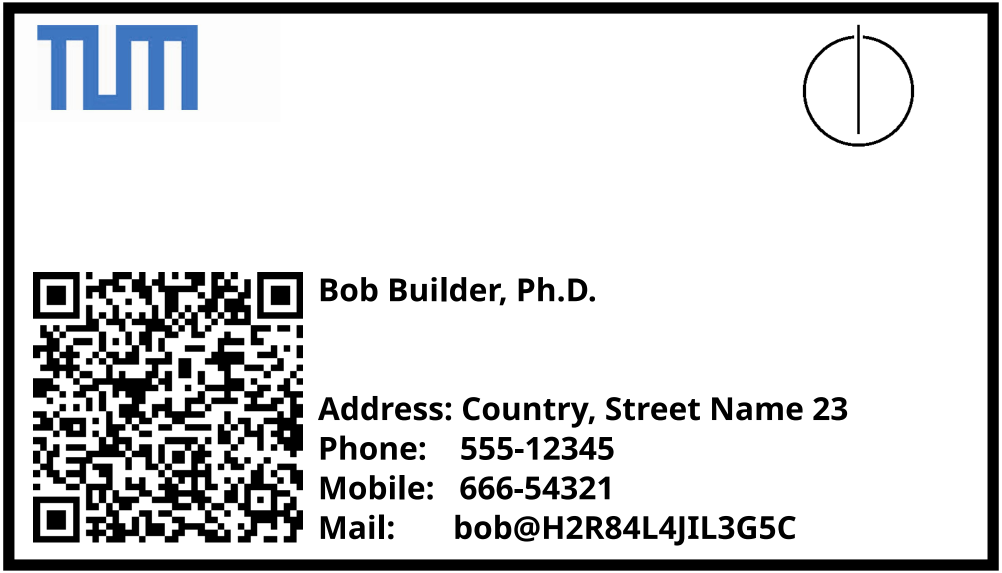
[✓ Master Zone] Private Zone Shorten Zone

Name	Type	Value	Expiration	Public
<new name>				
+	<new record>			
	MX	5,mail.+	end of time	✓
priv	<new record>			
	PKEY	3IQT1G601GUBVOS5C0JO87OEFB8N3DBJQ4L9SBI8PFLR8UKCVGHG	end of time	☐
heise	<new record>			
	LEHO	heise.de	end of time	✓
	AAAA	2a02:2e0:3fe:100::8	end of time	✓
	A	193.99.144.80	end of time	✓
home	<new record>			
大学	<new record>			
short	<new record>			
mail	<new record>			
homepage	<new record>			
fcfs	<new record>			
www	<new record>			

Name resolution in GNS



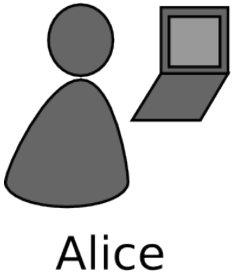
Secure introduction

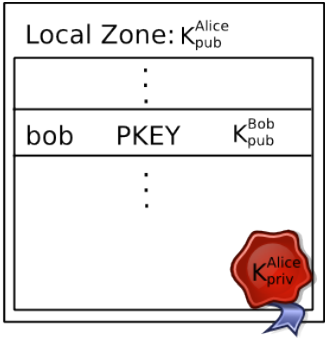


Bob Builder, Ph.D.

Address: Country, Street Name 23
Phone: 555-12345
Mobile: 666-54321
Mail: bob@H2R84L4JIL3G5C

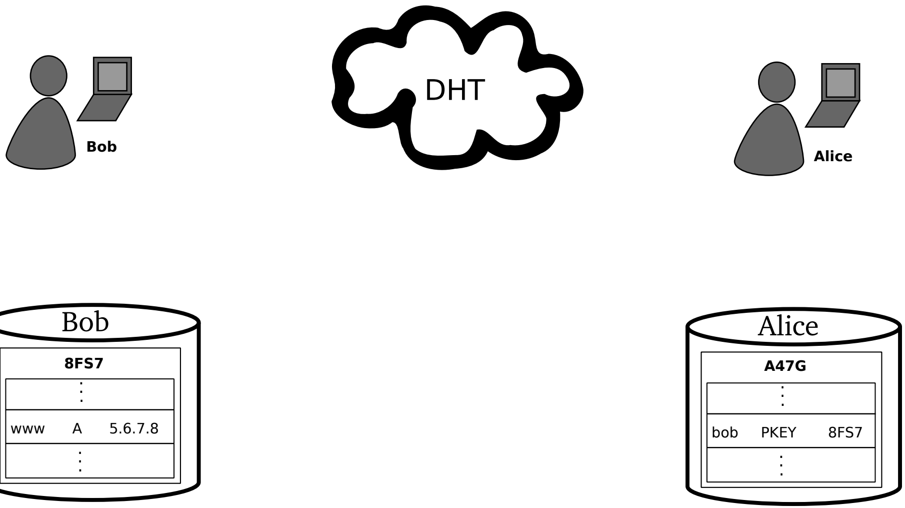
Delegation



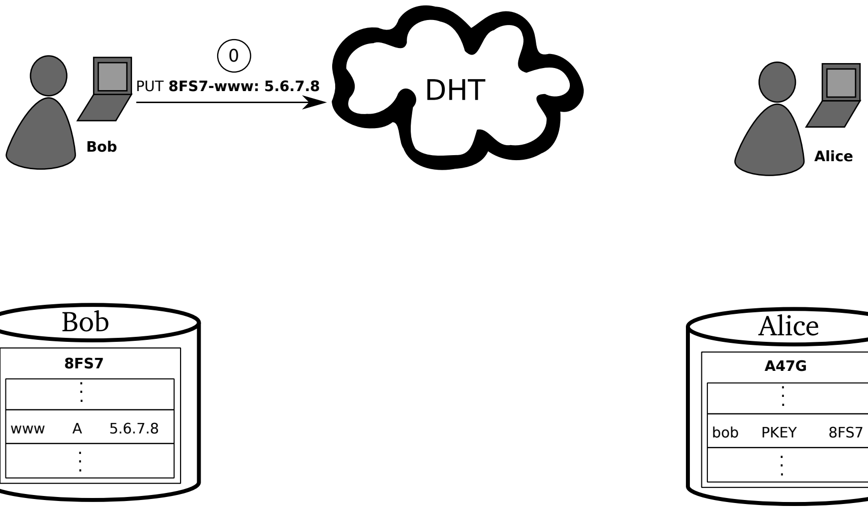


- Alice learns Bob's public key
- Alice creates delegation to zone K_{pub}^{Bob} under label **bob**
- Alice can reach Bob's webserver via **www.bob.gns.alt**

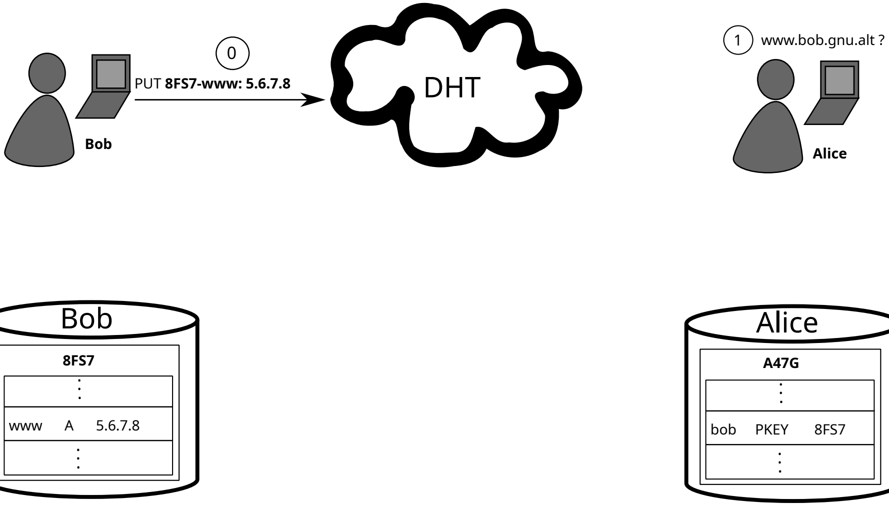
Name Resolution



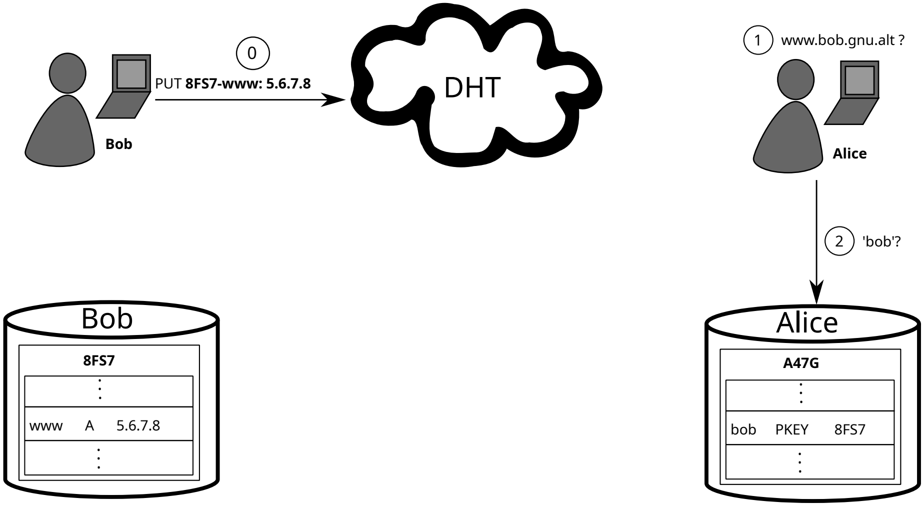
Name Resolution



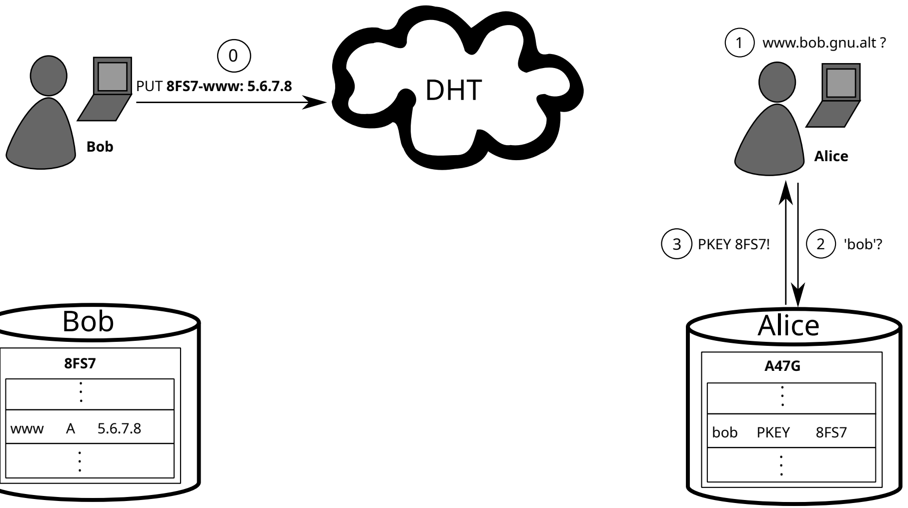
Name Resolution



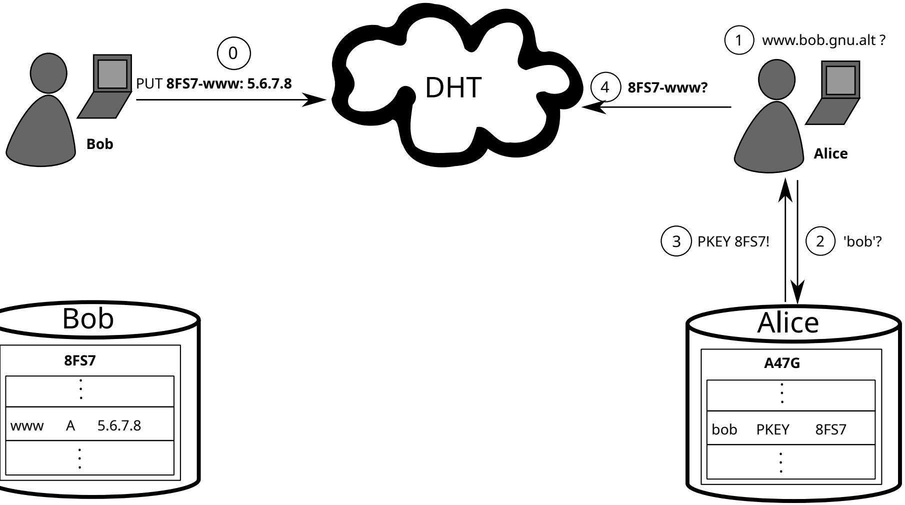
Name Resolution



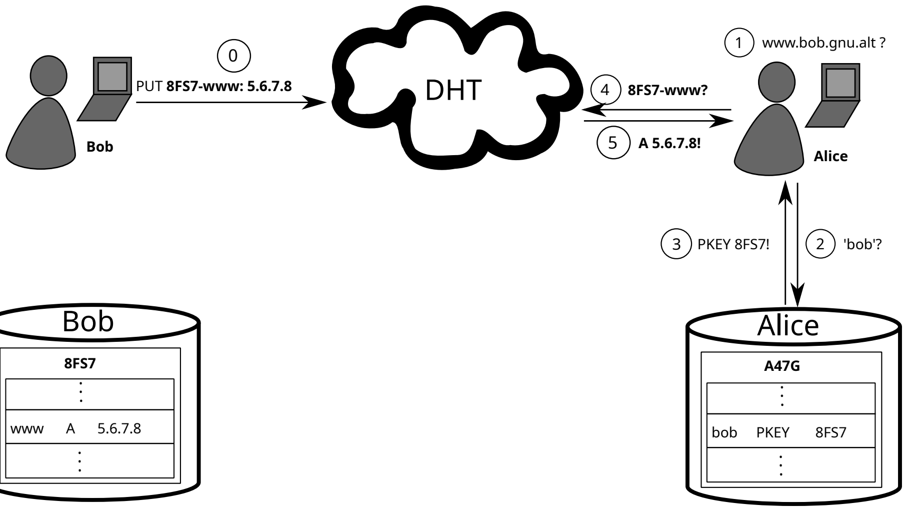
Name Resolution



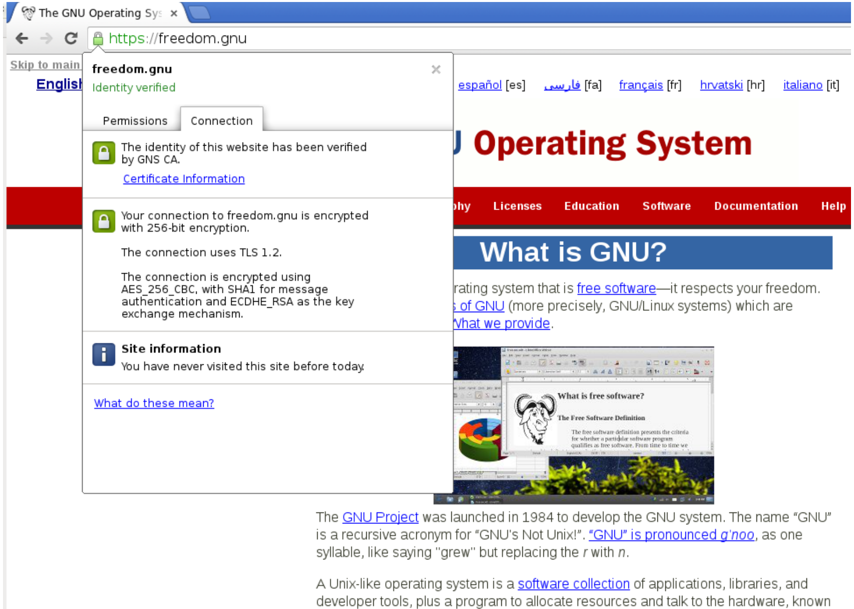
Name Resolution



Name Resolution



GNS as PKI (via DANE/TLSA)



The GNU Operating Sys ×

https://freedom.gnu

Skip to main

English

español [es] فارسی [fa] français [fr] hrvatski [hr] italiano [it]

freedom.gnu

Identity verified

Permissions | Connection

The identity of this website has been verified by GNS CA.

Certificate Information

Your connection to freedom.gnu is encrypted with 256-bit encryption.

The connection uses TLS 1.2.

The connection is encrypted using AES_256_CBC, with SHA1 for message authentication and ECDHE_RSA as the key exchange mechanism.

Site information

You have never visited this site before today.

What do these mean?

Operating System

hy Licenses Education Software Documentation Help

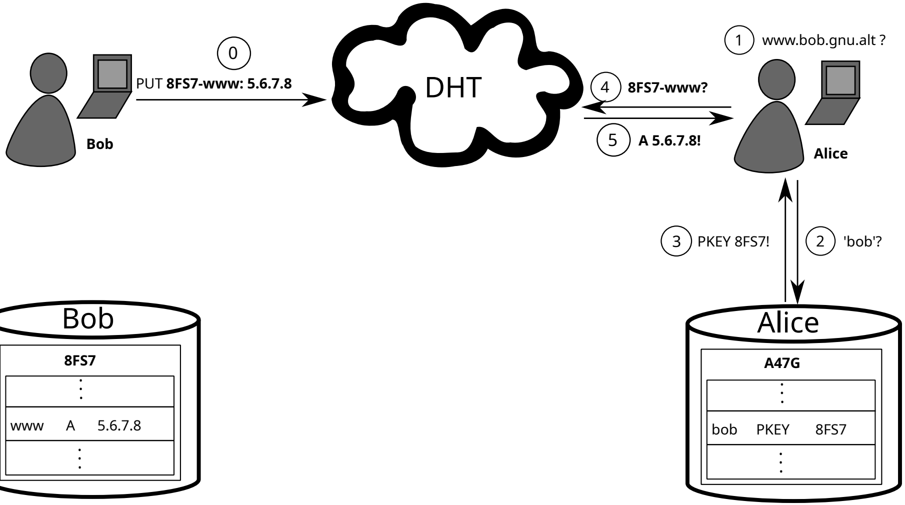
What is GNU?

rating system that is free software—it respects your freedom.
of GNU (more precisely, GNU/Linux systems) which are
What we provide.

The GNU Project was launched in 1984 to develop the GNU system. The name “GNU” is a recursive acronym for “GNU's Not Unix!”. “GNU” is pronounced *g'noo*, as one syllable, like saying "grew" but replacing the *r* with *n*.

A Unix-like operating system is a software collection of applications, libraries, and developer tools, plus a program to allocate resources and talk to the hardware, known

Privacy Issue: DHT



Query Privacy: Terminology

G	generator in ECC curve, a point
o	size of ECC group, $o := \|G\|$, o prime
x	private ECC key of zone ($x \in \mathbb{Z}_o$)
P	public key of zone, a point $P := xG$
l	label for record in a zone ($l \in \mathbb{Z}_o$)
$R_{P,l}$	set of records for label l in zone P
$q_{P,l}$	query hash (hash code for DHT lookup)
$B_{P,l}$	block with encrypted information for label l in zone P published in the DHT under $q_{P,l}$

Query Privacy: Cryptography

Publishing records $R_{P,l}$ as $B_{P,l}$ under key $q_{P,l}$

$$h := H(l, P) \tag{1}$$

$$d := h \cdot x \mod o \tag{2}$$

$$B_{P,l} := S_d(E_{HKDF(l,P)}(R_{P,l})), dG \tag{3}$$

$$q_{P,l} := H(dG) \tag{4}$$

Query Privacy: Cryptography

Publishing records $R_{P,l}$ as $B_{P,l}$ under key $q_{P,l}$

$$
\begin{aligned}
h &:= H(l, P) && (1)\\
d &:= h \cdot x \mod o && (2)\\
B_{P,l} &:= S_d(E_{HKDF(l,P)}(R_{P,l})), dG && (3)\\
q_{P,l} &:= H(dG) && (4)
\end{aligned}
$$

Searching for records under label l in zone P

$$
\begin{aligned}
h &:= H(l, P) && (5)\\
q_{P,l} &:= H(hP) = H(hxG) = H(dG) \Rightarrow \texttt{obtain } B_{P,l} && (6)\\
R_{P,l} &= D_{HKDF(l,P)}(B_{P,l}) && (7)
\end{aligned}
$$

Using cryptographic identifiers

- Zone are identified by a public key
- "alice.bob.*PUBLIC-KEY*" is perfectly legal in GNS!

$\Rightarrow$ Globally unique identifiers

Break

Key Revocation

- Certificate Revocation Lists (X.509)
- Online Certificate Status Protocol (OCSP)
- OCSP stapling (TLS)
- Publish revocation in blockchain?

Key Revocation

- Certificate Revocation Lists (X.509)
- Online Certificate Status Protocol (OCSP)
- OCSP stapling (TLS)
- Publish revocation in blockchain?
- Controlled flooding

Key Revocation via Controlled Flooding

- Revocation message signed with private key that is to be revoked
- Flooded on all links in (P2P) overlay, stored forever
- Expensive **proof-of-work** used to limit DoS-potential
- Proof-of-work can be calculated ahead of time
- Revocation messages can be computed and stored off-line if desired
- Efficient set reconciliation used when peers connect

Efficient Set Union

- Alice and Bob have sets A and B
- The sets are very large
- ... but their symmetric difference $\delta = |(A - B) \cup (B - A)|$ is small
- Now Alice wants to know $B - A$ (the elements she's missing)
- ... and Bob $A - B$ (the elements he's missing)
- How can Alice and Bob do this efficiently?
  - w.r.t. communication and computation

Simplistic Solution

- Naive approach: Alice sends A to Bob, Bob sends $B - A$ back to Alice
- ...and vice versa.

- Communication cost: $O(|A| + |B|)$ **:(**
- Ideally, we want to do it in $O(\delta)$.
- First improvement: Don't send elements of A and B, but send/request hashes. Still does not improve complexity **:(**

- We need some more fancy data structure!

Bloom Filters

Constant size data structure that “summarizes” a set.

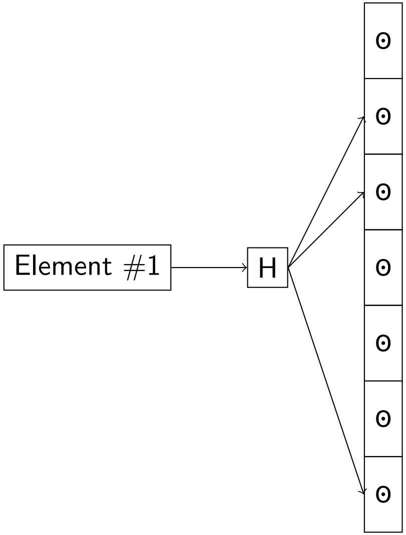
Operations:

$d = \mathit{NewBF}(\mathit{size})$ Create a new, empty bloom filter.

$\mathit{Insert}(d, e)$ Insert element e into the BF d.

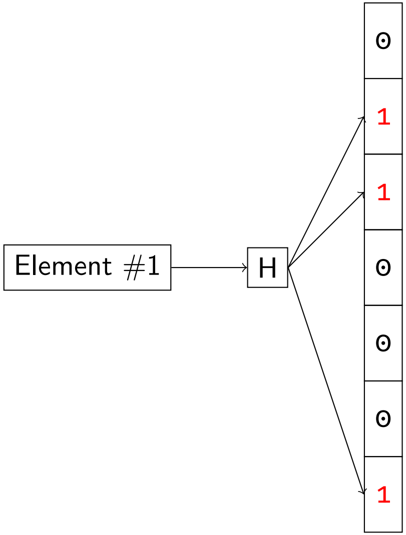
$b = \mathit{Contains}(d, e)$ Check if BF d contains element e.
$b \in \{\text{“Definitely not in set”}, \text{“Probably in set”}\}$

BF: Insert



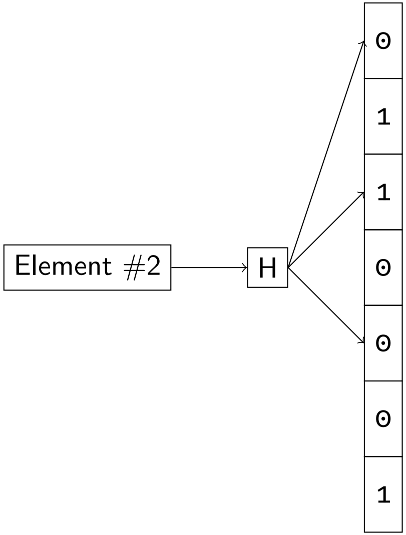
$H(\text{Element } \#1) = (2, 3, 7)$

BF: Insert



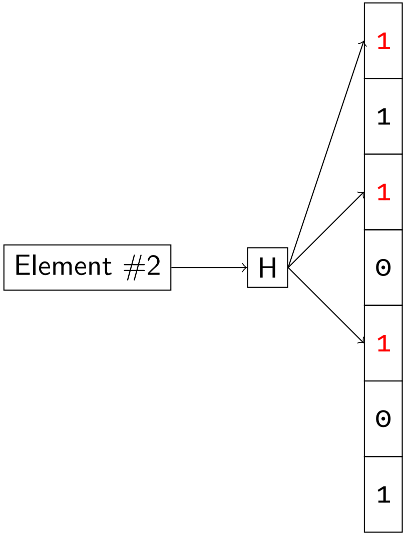
$H(\text{Element }\#1) = (2, 3, 7)$

BF: Insert



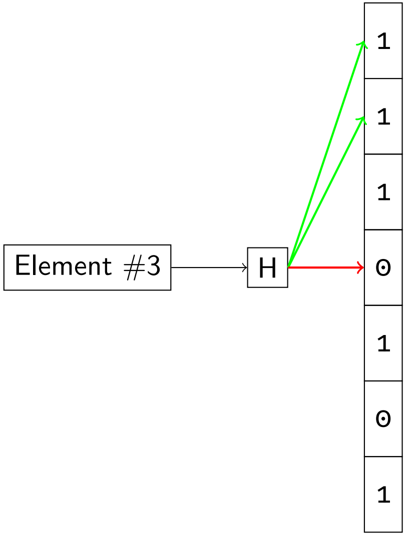
$$H(\text{Element } \#1) = (2, 3, 7)$$
$$H(\text{Element } \#2) = (1, 3, 5)$$

BF: Insert



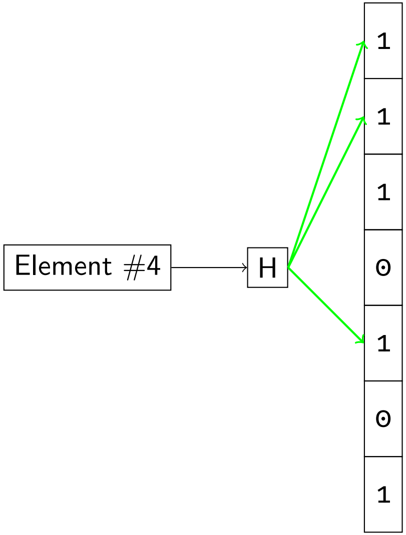
$$H(\text{Element } \#1) = (2, 3, 7)$$
$$H(\text{Element } \#2) = (1, 3, 5)$$

BF: Membership Test



$$H(\text{Element }\#1) = (2, 3, 7)$$
$$H(\text{Element }\#2) = (1, 3, 5)$$

BF: Membership Test (false positive)



$H(\text{Element }\#1) = (2, 3, 7)$
$H(\text{Element }\#2) = (1, 3, 5)$

Counting Bloom Filters

BF where buckets hold a **positive integer**.

Additional Operation:

Remove(d, e) Remove element from the CBF d.

$\Rightarrow$ False negatives when removing a non-existing element.

Invertible Bloom Filters

Similar to CBF, but

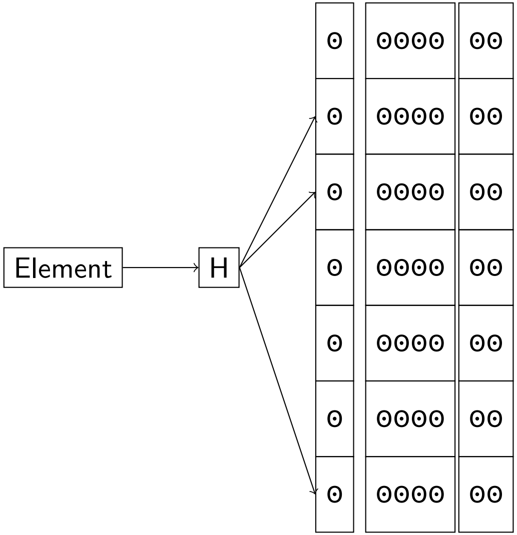
- Allow **negative counts**
- Additionaly store **(XOR-)sum of IDs (IDSUM)** in each bucket.
- Additionaly store **(XOR-)sum of hashes (XHASH)** in each bucket.

Additional Operations:

$(e, r) = \mathit{Extract}(d)$ Extract an element ID (e) from the IBF d, with result code $r \in \{\mathit{left}, \mathit{right}, \mathit{done}, \mathit{fail}\}$

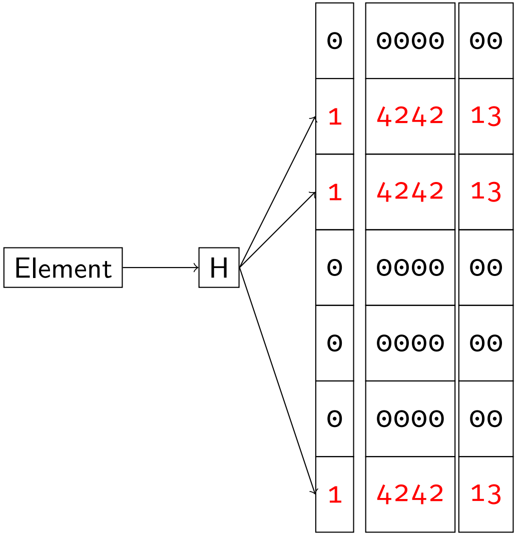
$d' = \mathit{SymDiff}(d_1, d_2)$ Create an IBF that represents the symmetric difference of d_1 and d_2.

IBF: Insert Element #1



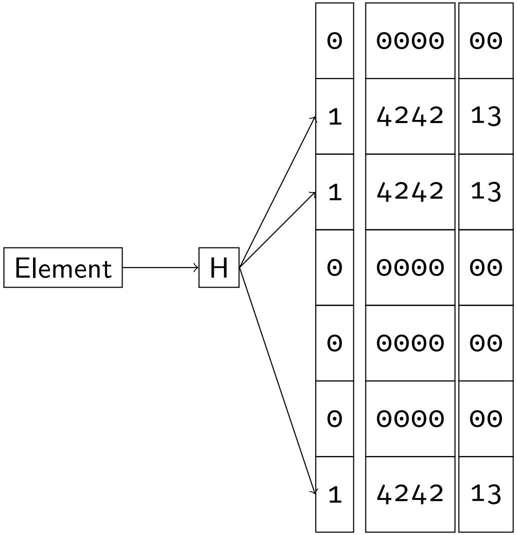
$H(\text{Element } \#1) \mapsto (2, 3, 7)$
$H'(\text{Element } \#1) \mapsto 4242 \text{ (ID)}$
$H''(4242) \mapsto 13$

IBF: Insert Element #1



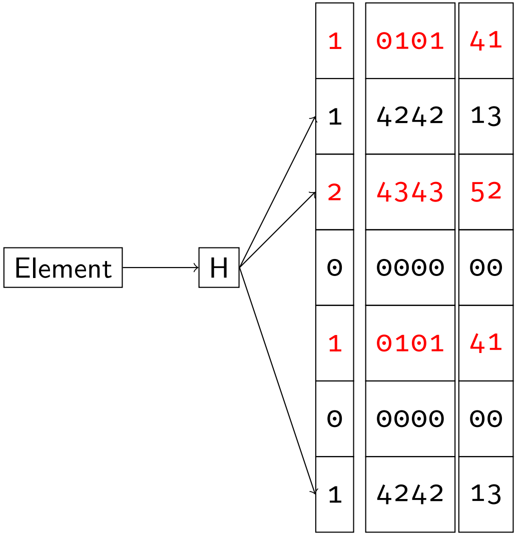
$H(\text{Element } \#1) \mapsto (2, 3, 7)$
$H'(\text{Element } \#1) \mapsto 4242 \text{ (ID)}$
$H''(4242) \mapsto 13$

IBF: Insert Element #2



$H(\text{Element } \#2) = (1, 3, 5)$
$H'(\text{Element } \#2) = 0101 \text{ (ID)}$
$H''(0101) \mapsto 41$

IBF: Insert Element #2



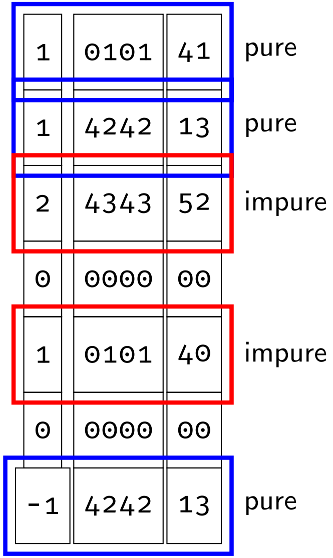
$H(\text{Element } \#2) = (1, 3, 5)$
$H'(\text{Element } \#2) = 0101 \text{ (ID)}$
$H''(0101) \mapsto 41$

Symmetric Difference on IBFs

We can directly compute the symmetric difference without extraction.

- Subtract counters
- XOR of IDSUM and XHASH values

IBF: Extract



- $|counter| = 1 \wedge H''(IDSUM) = XHASH \Leftrightarrow$ `pure`
- Impure bucket $\Rightarrow$ potential decoding failure
- Pure bucket $\Rightarrow$ extractable element ID
- Extraction $\Rightarrow$ more pure buckets (hopefully/probably)
- Less elements $\Rightarrow$ more chance for pure buckets

The Set Union Protocol [7]

1. Create IBFs
2. Compute SymDiff
3. Extract element IDs

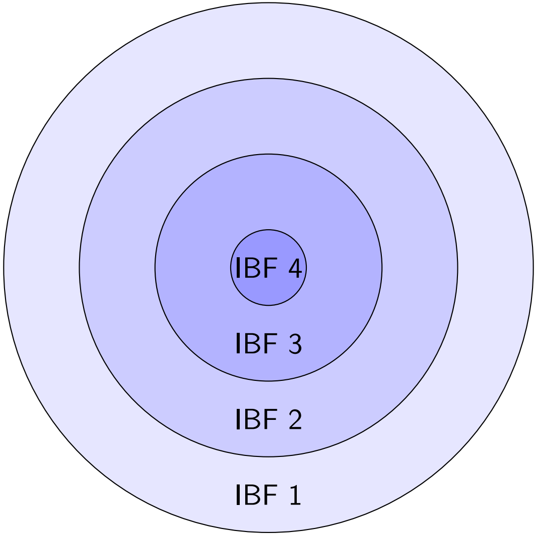
- Amount of communication and computation only depends on δ, not $|A| + |B|$:)
- How do we choose the initial size of the IBF?
- $\Rightarrow$ Do difference estimation first!

Difference Estimation

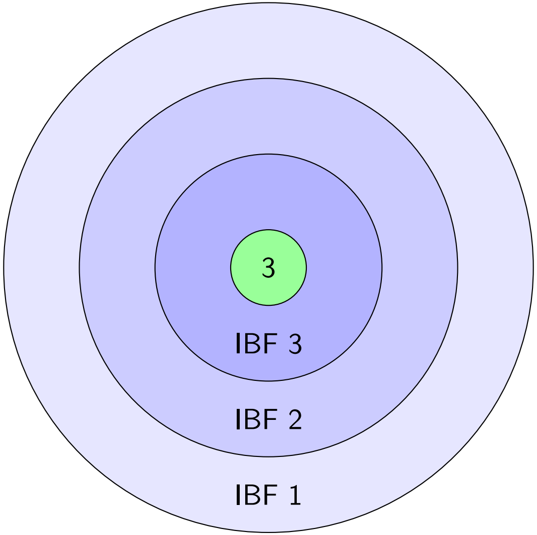
- We need an estimator that's accurate for small differences
- Turns out we can re-use IBFs for difference estimation:

1. Alice and Bob create fixed number of constant-size IBFs by sampling their set. The collection of IBFs is called a Strata Estimator (SE).
    - Stratum 1 contains 1/2 of all elements
    - Stratum 2 contains 1/4 of all elements
    - Stratum n contains $1/(2^n)$ all elements
2. Alice receives Bob's strata estimator
3. Alice computes $SE_{diff} = SymDiff(SE_{\mathrm{Alice}}, SE_{\mathrm{Bob}})$
    - by pair-wise *SymDiff* of all IBFs in the SE
4. Alice estimates the size of SE_{diff}.

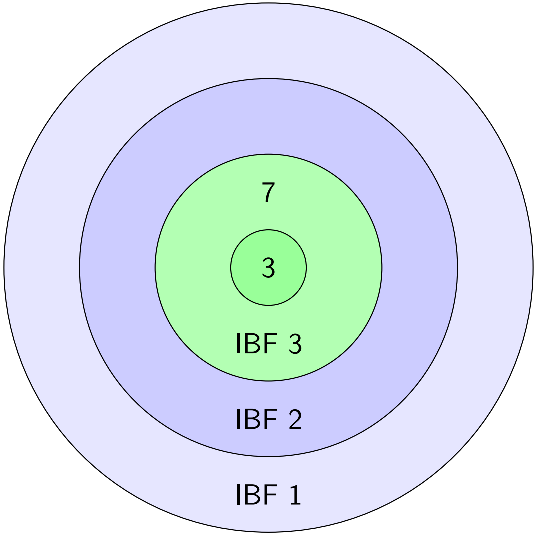
Strata Estimator



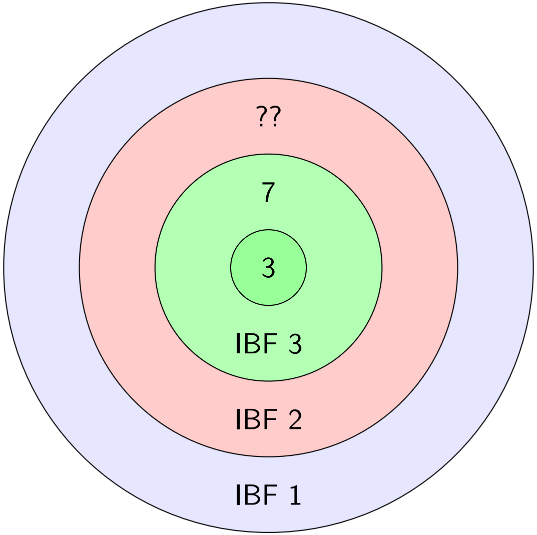
Strata Estimator



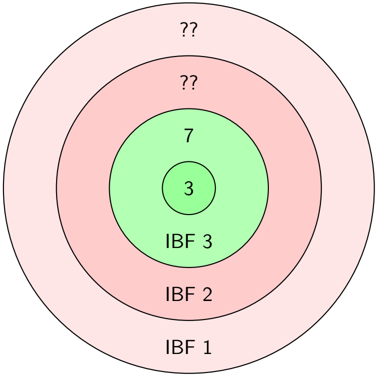
Strata Estimator



Strata Estimator



Estimation

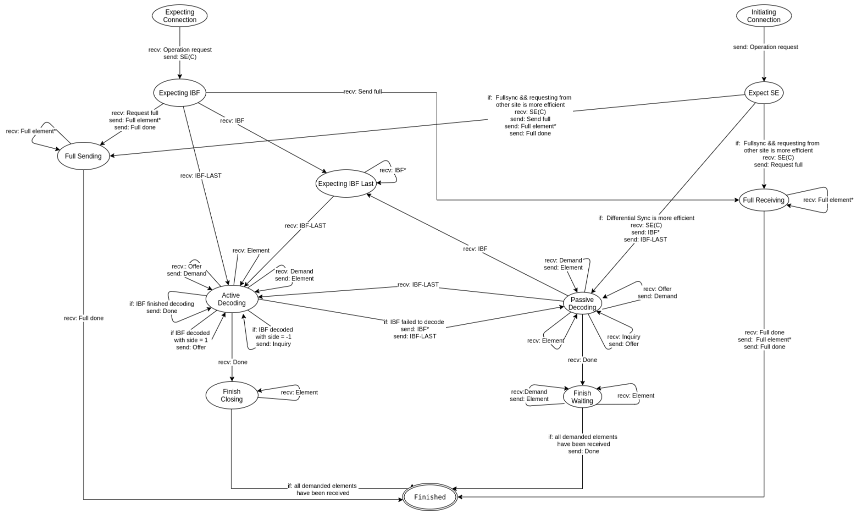


Estimate set size difference as $\frac{2^4 \cdot 3 + 2^3 \cdot 7}{2}$.

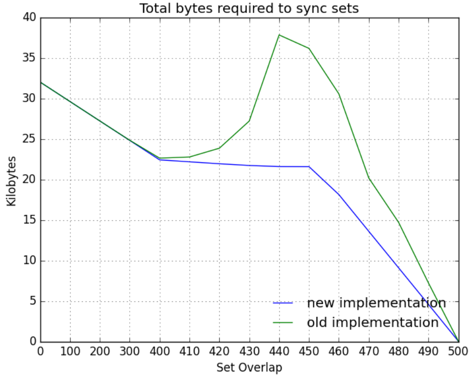
The naïve IBF Protocol

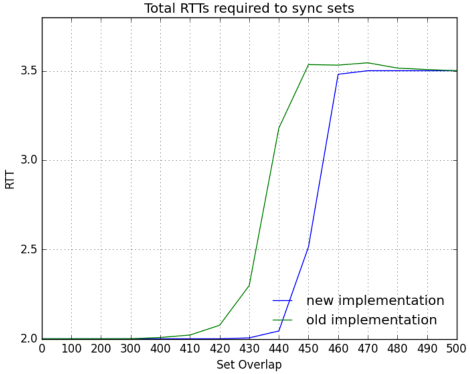
1. Alice sends SE_{Alice} to Bob
2. Bob estimates the set difference δ
3. Bob computes IBF_{Bob} for size δ and sends it to Alice
4. Alice computes IBF_{Alice}
5. Alice computes $IBF_{diff} = SymDiff(IBF_{\text{Alice}}, IBF_{\text{Bob}})$
6. Alice extracts element IDs from IBF_{diff}.
    - $b = left \Rightarrow$ Send element to to Bob
    - $b = right \Rightarrow$ Send element request to to Bob
    - $b = fail \Rightarrow$ Send larger IBF (double the size) to Bob, go to (3.) with switched roles
    - $b = done \Rightarrow$ We're done ...

The Complete Protocol



Implementation Performance: Tuning required!





GNS Summary

- Interoperable with DNS
- Globally unique identifiers with ".PUBLIC-KEY"
- Delegation allows using zones of other users
- Trust paths explicit, trust agility
- Simplified key exchange compared to Web-of-Trust
- Privacy-enhanced queries, censorship-resistant
- Reliable revocation using flooding with proof-of-work

Privacy summary

Method	Defense against MiTM	Zone privacy	Privacy vs. network	Privacy vs. operator	Traffic amplification resistance	Censorship resistance	Ease of migration
DNS	✗	✓	✗	✗	✗	✗	✓
DNSSEC	✓	✗	✗	✗	✗	✗	✗*
DNSCurve	✓	✓	✓	✗	✓	✗	✗
DNS-over-TLS	✓	n/a	✓	✗	✓	✗	✗
Namecoin	✓	✗	✓	✓	✓	✓	✗
RAINS	✓	✗	✓	✗	✓	✗	✗
GNS	✓	✓	✓	✓	✓	✓	✗

*EDNS0

Key management summary

	Suitable for personal use	Memorable	Decentralised	Modern cryptography	Understandable	Exposes metadata	Transitive
DNS	✗	✓	✗	✗	✗	✗	✓
DNSSEC	✗	✓	✗	✗	✗	✗	✓
DNSCurve	✗	✓	✗	✓	✗	✗	✓
DNS-over-TLS	✗	✓	✗	✗	✗	✗	✓
TLS-X.509	✗	✓	✗	✗	✗	✗	✓
Web of Trust	✓	✗	✓	✗	✗	✗	✓
TOFU	✓	✗	✓		✓	✓	✗
Namecoin	✗	✓	✗	✓	✓	✗	✓
RAINS	✗	✓	✗	✓	✓	✗	✓
GNS	✓	✓	✓	✓	✓	✓	✓

Case study: GNS

DNS is known to suffer from a lack of end-to-end integrity protections. As a result, Chinese "great firewall" DNS manipulation has been shown to impact name resolution even in Europe.

> *"The GNU Name System (GNS) establishes a new name system using cryptography where zone data, queries and replies are private. The use of a distributed hash table (DHT) implies that resolution costs are comparable to those of DNS. However, states and ISPs cannot monitor or block queries, limiting their ability to protect the public from malicious Web sites. Names are not globally unique, allowing multiple anonymous users to lay claim to the same name. However, the system includes some well-known mappings by default, which users are unlikely to change. Trademarks, copyrights anti-fraud or anti-terrorism judgements can only be enforced against those well-known mappings, which users are able to bypass."*

Discuss virtues and vices affected.

Conclusion

DNS	globalist
DNSSEC	authoritarian
Namecoin	libertarian (US)
RAINS	nationalist
GNS	anarchist

In which world do you want to live?

Part VI: Introduction to GNUnet

Internet Design Goals, David Clark, 1988

1. **Internet communication must continue despite loss of networks or gateways.**
2. The Internet must support multiple types of communications service.
3. The Internet architecture must accommodate a variety of networks.
4. The Internet architecture must permit *distributed management* of its resources.
5. The Internet architecture must be cost effective.
6. The Internet architecture must permit host attachment with a low level of effort.
7. **The resources used in the internet architecture must be accountable.**

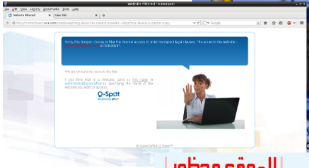
Where We Are

Where We Are

Source: csmoni

Source: gawand.org

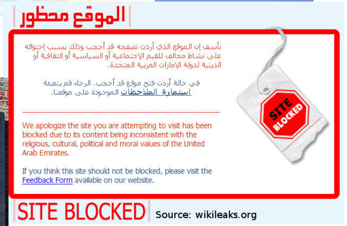
Source: padresteve.com



Sorry, this hotspot choose to filter the internet access in order to respect legal clauses. The acces to the website : motherboard.vice.com is forbidden*.

*We did not found the reason to this filter

If you think that is a mistake, send us this page to webmaster@spotcoffee.eu specifying the name of the website you want to access.


الموقع محظور

نأسف إن الموقع الذي أردت تصفحه قد أحجب وذلك بسبب إحتوائه على نشاط مخالف للقيم الإجتماعية أو السياسية أو الثقافية أو الدينية لدولة الإمارات العربية المتحدة.

في حالة أردت فتح موقع قد أحجب. الرجاء قم بتعبئة استمارة الملاحظات الموجودة على موقعنا.

We apologize the site you are attempting to visit has been blocked due to its content being inconsistent with the religious, cultural, political and moral values of the United Arab Emirates.

If you think this site should not be blocked, please visit the Feedback Form available on our website.

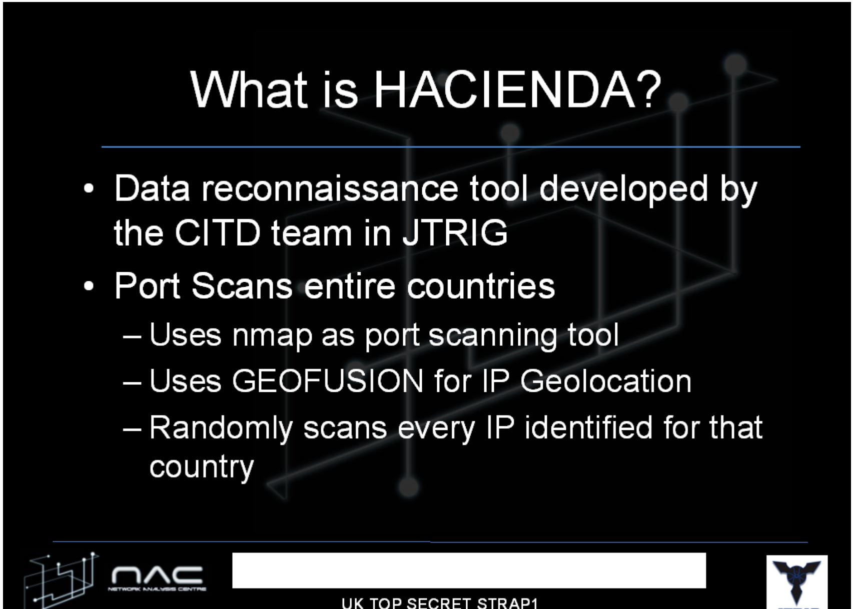
SITE BLOCKED

Source: wikileaks.org

Source: wikicommons.org

Source: enturebeat.com

Example 1: Collateral Damage



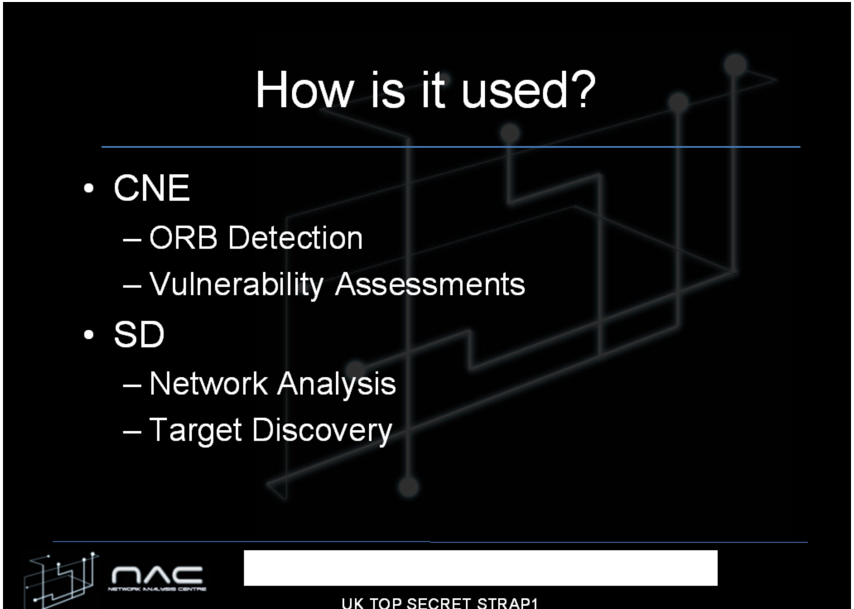
What is HACIENDA?

- Data reconnaissance tool developed by the CITD team in JTRIG
- Port Scans entire countries
  - Uses nmap as port scanning tool
  - Uses GEOFUSION for IP Geolocation
  - Randomly scans every IP identified for that country

NAC NETWORK ANALYSIS CENTRE

UK TOP SECRET STRAP1

Example 1: Collateral Damage



How is it used?

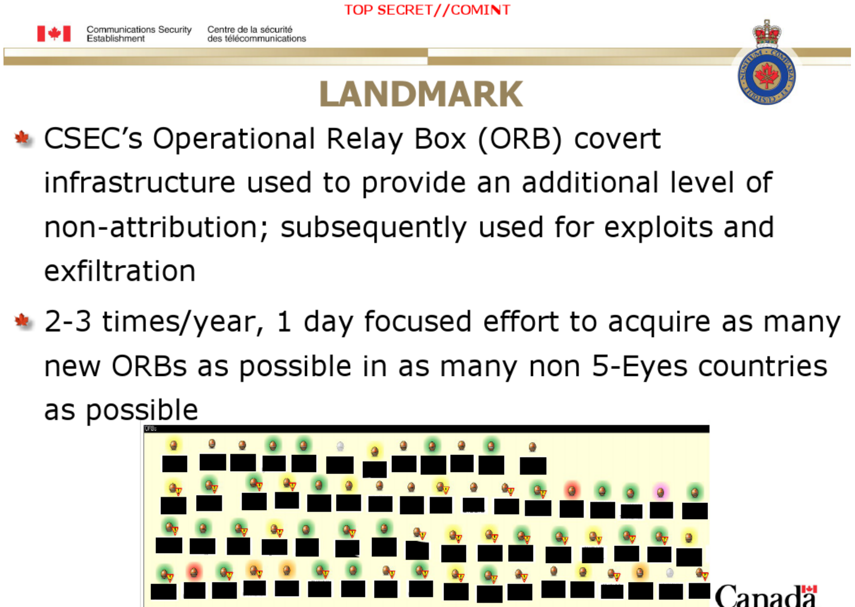
- CNE
  - ORB Detection
  - Vulnerability Assessments
- SD
  - Network Analysis
  - Target Discovery

NAC NETWORK ANALYSIS CENTRE

[redacted]

UK TOP SECRET STRAP1

Example 1: Collateral Damage



TOP SECRET//COMINT

Communications Security Establishment — Centre de la sécurité des télécommunications

LANDMARK

- CSEC's Operational Relay Box (ORB) covert infrastructure used to provide an additional level of non-attribution; subsequently used for exploits and exfiltration
- 2-3 times/year, 1 day focused effort to acquire as many new ORBs as possible in as many non 5-Eyes countries as possible

ORBs:

[redacted] [redacted] [redacted] [redacted] [redacted] [redacted] [redacted] [redacted] [redacted] [redacted] [redacted] [redacted]

[redacted] [redacted] [redacted] [redacted] [redacted] [redacted] [redacted] [redacted] [redacted] [redacted] [redacted] [redacted] [redacted] [redacted] [redacted] [redacted] [redacted]

[redacted] [redacted] [redacted] [redacted] [redacted] [redacted] [redacted] [redacted] [redacted] [redacted] [redacted] [redacted] [redacted] [redacted] [redacted] [redacted]

[redacted] [redacted] [redacted] [redacted] [redacted] [redacted] [redacted] [redacted] [redacted] [redacted] [redacted] [redacted] [redacted] [redacted] [redacted] [redacted] [redacted] [redacted]

Canada

Why should you care?

If you are …

- … of any importance in the world, or
- … a system or network administrator, or
- … a security researcher, or
- … in this room, or
- … mistaken for any of the above,

Why should you care?

If you are ...

- ... of any importance in the world, or
- ... a system or network administrator, or
- ... a security researcher, or
- ... in this room, or
- ... mistaken for any of the above,

then you are probably a target.

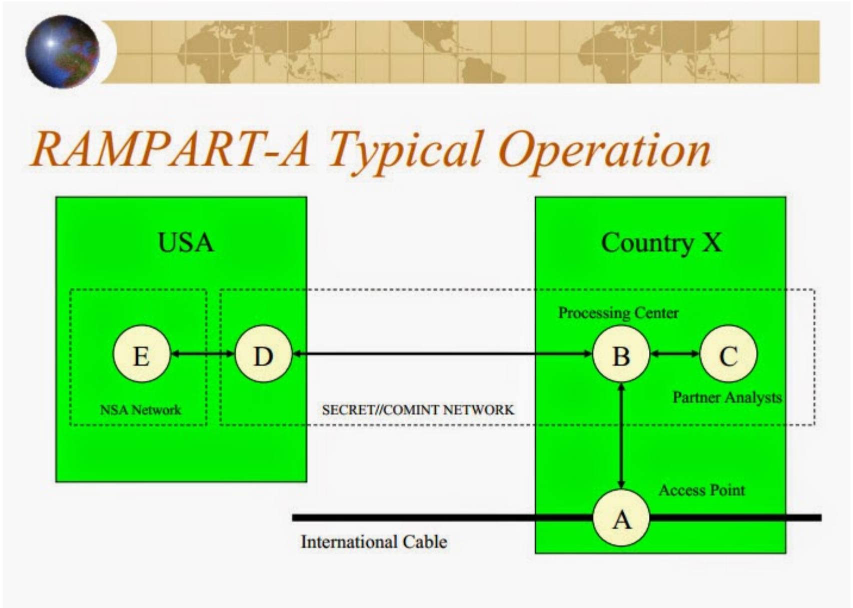
So what if they listen to my calls?

- Kompromat — and you do not get to decide what is bad!
- Self-censorship
- Loss of business
- No privacy $\Rightarrow$ No free press $\Rightarrow$ No liberal democracy

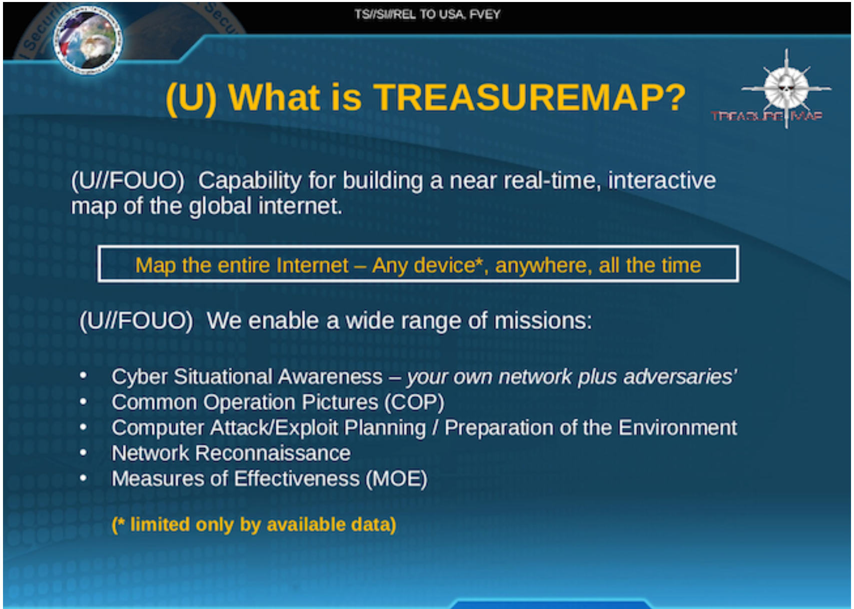
So what if they listen to my calls?

- Kompromat — and you do not get to decide what is bad!
- Self-censorship
- Loss of business
- No privacy $\Rightarrow$ No free press $\Rightarrow$ No liberal democracy
- Security services also get you drunk, encourage you to drive, arrest you for drunken driving and then ask you for your customer data.

Example 2: Owning the Network



Example 2: Owning the Network



TS//SI//REL TO USA, FVEY

(U) What is TREASUREMAP?

TREASURE MAP

(U//FOUO) Capability for building a near real-time, interactive map of the global internet.

Map the entire Internet – Any device*, anywhere, all the time

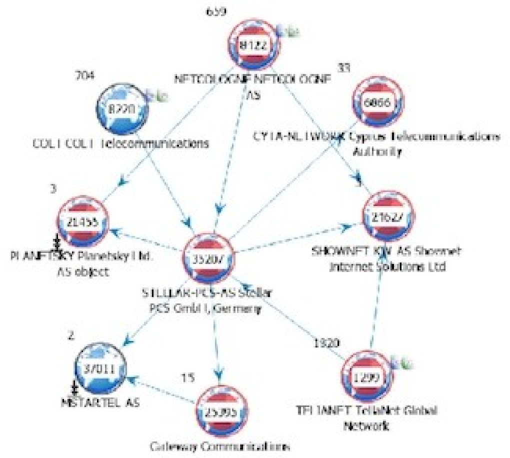
(U//FOUO) We enable a wide range of missions:

- Cyber Situational Awareness – *your own network plus adversaries'*
- Common Operation Pictures (COP)
- Computer Attack/Exploit Planning / Preparation of the Environment
- Network Reconnaissance
- Measures of Effectiveness (MOE)

(* limited only by available data)

Example 2: Owning the Network (Video)

TOP SECRET STRAP1


659
8122
NETCOLOGNE NETCOLOGNE AS
704
8220
COLT COLT Telecommunications
33
6866
CYTA-NETWORK Cyprus Telecommunications Authority
3
21455
PLANETSKY Planetsky Ltd. AS object
35207
STELLAR-PCS-AS Stellar PCS GmbH, Germany
21627
SHOWNET-KW AS Shownet Internet Solutions Ltd
2
37011
MSTARTEL AS
15
25395
Gateway Communications
1820
1299
TELIANET TeliaNet Global Network

Generated via TeasureMap

The Internet is Broken

Administrators have power.

Power attracts Mexican drug cartels.

Adversary model: Mexican drug cartel

- They took your family, and will brutally kill them if you do not give them what they want.
- Under these circumstances, you must still not be able to assist, and the public system design must make that clear.
- Thus, the cartel has nothing to gain from abducting your family and will not bother with it.

System administrators are targets of such an adversary.

Design Choices for a Civil Network!

Internet Design Goals (David Clark, 1988)

1. **Internet communication must continue despite loss of networks or gateways.**
2. The Internet must support multiple types of communications service.
3. The Internet architecture must accommodate a variety of networks.
4. The Internet architecture must permit *distributed management* of its resources.
5. The Internet architecture must be cost effective.
6. The Internet architecture must permit host attachment with a low level of effort.
7. **The resources used in the internet architecture must be accountable.**

GNUnet Design Goals

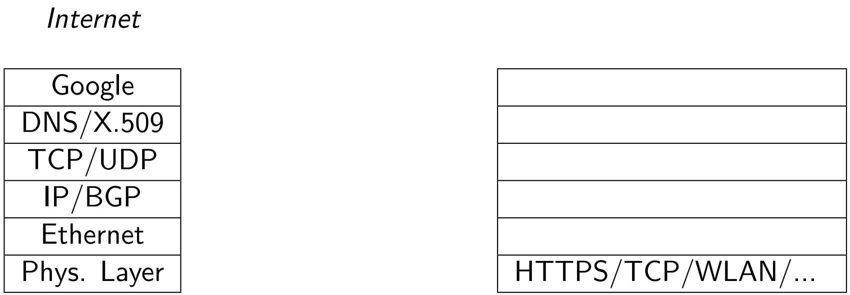
1. GNUnet must be implemented as free software.
2. **The GNUnet must only disclose the minimal amount of information necessary.**
3. **The GNUnet must be decentralised and survive Byzantine failures in any position in the network.**
4. **The GNUnet must make it explicit to the user which entities must be trustworthy when establishing secured communications.**
5. **The GNUnet must use compartmentalization to protect sensitive information.**
6. The GNUnet must be open and permit new peers to join.
7. **The GNUnet must be self-organizing and not depend on administrators.**
8. The GNUnet must support a diverse range of applications and devices.
9. The GNUnet architecture must be cost effective.
10. **The GNUnet must provide incentives for peers to contribute more resources than they consume.**

Let's Implement It!

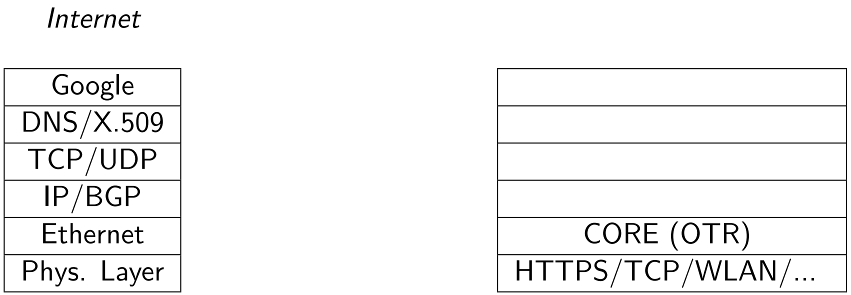
Internet

Google
DNS/X.509
TCP/UDP
IP/BGP
Ethernet
Phys. Layer

Let's Implement It!



Let's Implement It!



Let's Implement It!

Internet

Google
DNS/X.509
TCP/UDP
IP/BGP
Ethernet
Phys. Layer

R^5N DHT
CORE (OTR)
HTTPS/TCP/WLAN/...

Let's Implement It!

Internet

Google
DNS/X.509
TCP/UDP
IP/BGP
Ethernet
Phys. Layer

CADET (Axolotl+SCTP)
R^5N DHT
CORE (OTR)
HTTPS/TCP/WLAN/...

Let's Implement It!

Internet	
Google	
DNS/X.509	GNU Name System
TCP/UDP	CADET (Axolotl+SCTP)
IP/BGP	R^5N DHT
Ethernet	CORE (OTR)
Phys. Layer	HTTPS/TCP/WLAN/...

Let's Implement It!

Internet

Google
DNS/X.509
TCP/UDP
IP/BGP
Ethernet
Phys. Layer

Applications
GNU Name System
CADET (Axolotl+SCTP)
R^5N DHT
CORE (OTR)
HTTPS/TCP/WLAN/...

Let’s Implement It!

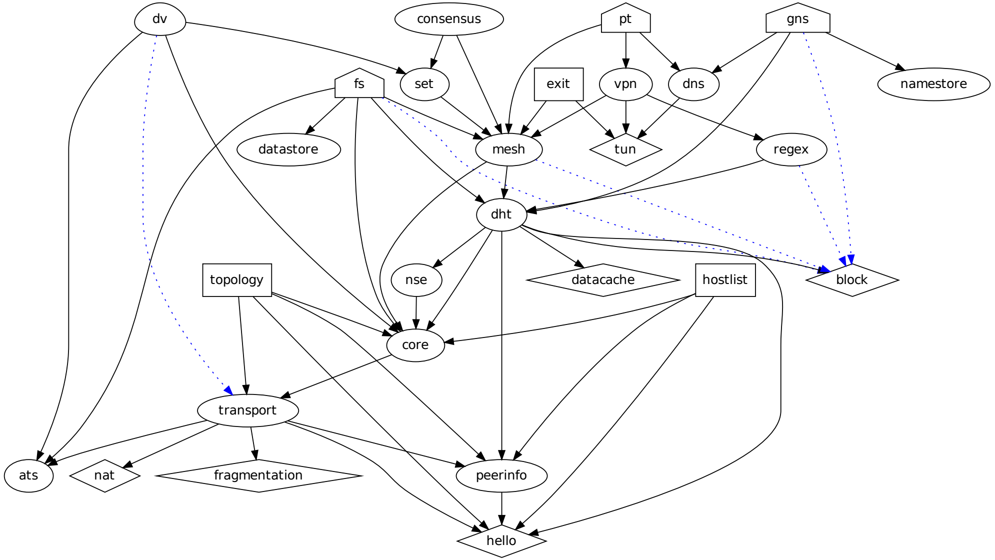
Internet

Google
DNS/X.509
TCP/UDP
IP/BGP
Ethernet
Phys. Layer

GNUnet

Applications
GNU Name System
CADET (Axolotl+SCTP)
R^5N DHT
CORE (OTR)
HTTPS/TCP/WLAN/...

A real peer: Dependencies



A GNUnet Service is a Process

- If all subsystems are used, GNUnet would currently use ≈ 40 processes (services and daemons)
- user interfaces increase this number further
- systemd-like `gnunet-service-arm` starts them
- services are manipulated using the respective command-line tool

$\Rightarrow$ `gnunet-arm -s` starts GNUnet

Applications (being) built using GNUnet

- Anonymous and non-anonymous file-sharing
- IPv6–IPv4 protocol translator and tunnel
- GNU Name System: censorship-resistant replacement for DNS
- Conversation: secure, decentralised VoIP
- SecuShare, a social networking application
- GNU Taler: privacy-preserving payments
- ...

A Pattern of Hope

Spy Program	Target	Defense	Started
FTM/TRACFIN	SWIFT/VISA/etc.	DigiCash/GNU Taler	1990
TREASUREMAP	Internet (all)	Freenet/GNUnet/Tor	2000
HACIENDA	vuln. TCP service	Port Knocking	2000
BULLRUN/DUAL_EC_DRBG	PRNG (backdoor)	n/a	2004
BULLRUN/LONGHAUL	TLS/IPSEC (keys)	OTR/AXOLOTL	2004
MJOLNIR	Long-path in Tor	Tor 0.2.3.11	2007
PRISM	US big data corps	SecuShare	2009
MORECOWBELL	DNS	GNU Name System	2012
...	...	...	...

Exercise

```
# apt-get install git autoconf automake autopoint gettext
# apt-get install libunistring-dev libgnutls28-dev
# apt-get install openssl gnutls-bin libtool libltdl-dev
# apt-get install libcurl-gnutls-dev libidn11-dev
# apt-get install libsqlite3-dev libjansson-dev libpq-dev
$ git clone git://git.gnunet.org/libmicrohttpd
$ git clone git://git.gnunet.org/gnunet
$ git clone git://git.gnunet.org/gnunet-gtk
$ for n in libmicrohttpd gnunet gnunet-gtk do;
    cd $n ; ./bootstrap ; ./configure --prefix=$HOME ...
    make install
    cd ..
  done
```

Exercise

```
$ gnunet-arm -s # launch peer
$ gnunet-namestore-gtk # configure your GNS zone
$ gnunet-gns # command-line resolution
$ gnunet-gns-proxy # launch SOCKS proxy
$ firefox # configure browser to use proxy
```

Break

Part V: Advanced Cryptographic Primitives

Homomorphic Encryption

$$E(x_1 \oplus x_2) = E(x_1) \otimes E(x_2) \tag{8}$$

Multiplicative Homomorphism: RSA & ElGamal

- Unpadded RSA (multiplicative):

$$E(x_1) \cdot E(x_2) = x_1^e x_2^e = E(x_1 \cdot x_2) \tag{9}$$

- ElGamal:

$$\begin{aligned} E(x_1) \cdot E(x_2) &= (g^{r_1}, x_1 \cdot h^{r_1})(g^{r_2}, x_2 \cdot h^{r_2}) && (10) \\ &= (g^{r_1+r_2}), (x_1 \cdot x_2)h^{r_1+r_2}) && (11) \\ &= E(x_1 \cdot x_2) && (12) \end{aligned}$$

Additive Homomorphism: Paillier

$$\mathrm{E}_K(m) := g^m \cdot r^n \mod n^2, \tag{13}$$

$$\mathrm{D}_K(c) := \frac{(c^\lambda \mod n^2) - 1}{n} \cdot \mu \mod n \tag{14}$$

where the public key $K = (n, g)$, m is the plaintext, c the ciphertext, n the product of $p, q \in \mathbb{P}$ of equal length, and $g \in \mathbb{Z}^*_{n^2}$. In Paillier, the private key is (λ, μ), which is computed from p and q as follows:

$$\lambda := \mathrm{lcm}(p-1, q-1), \tag{15}$$

$$\mu := \left(\frac{(g^\lambda \mod n^2) - 1}{n}\right)^{-1} \mod n. \tag{16}$$

Paillier offers additive homomorphic public-key encryption, that is:

$$\mathrm{E}_K(a) \otimes \mathrm{E}_K(b) \equiv \mathrm{E}_K(a+b) \tag{17}$$

for any public key K.

Fully homomorphic encryption

Additive:

$$E(A) \oplus E(B) = E(A + B) \tag{18}$$

and multiplicative:

$$E(A) \otimes E(B) = E(A \cdot B) \tag{19}$$

Known cryptosystems: Brakerski-Gentry-Vaikuntanathan (BGV), NTRU, Gentry-Sahai-Waters (GSW).

Pairing-based cryptography

Let G_1, G_2 be two additive cyclic groups of prime order q, and G_T another cyclic group of order q (written multiplicatively). A pairing is an efficiently computable map e:

$$e : G_1 \times G_2 \to G_T \tag{20}$$

which satisfies $e \neq 1$ and bilinearity:

$$\forall_{a,b \in F_q^*},\ \forall_{P \in G_1, Q \in G_2} :\ e\left(aP, bQ\right) = e\left(P, Q\right)^{ab} \tag{21}$$

Examples: Weil pairing, Tate pairing.

Hardness assumption

Computational Diffie Hellman:

$$g, g^x, g^y \Rightarrow g^{xy} \tag{22}$$

remains hard on G even given e.

Boneh-Lynn-Sacham (BLS) signatures [2]

Key generation:

Pick random $x \in \mathbb{Z}_q$

Signing:

$\sigma := h^x$ where $h := H(m)$

Verification:

Given public key g^x:

$$e(\sigma, g) = e(h, g^x) \tag{23}$$

Boneh-Lynn-Sacham (BLS) signatures [2]

Key generation:
: Pick random $x \in \mathbb{Z}_q$

Signing:
: $\sigma := h^x$ where $h := H(m)$

Verification:
: Given public key g^x:

$$e(\sigma, g) = e(h, g^x) \tag{23}$$

Why:

$$e(\sigma, g) = e(h, g)^x = e(h, g^x) \tag{24}$$

due to bilinearity.

Fun with BLS

Given signature $\langle \sigma, g^x \rangle$ on message h, we can *blind* the signature and public key g^x:

$$e(\sigma^b, g) = e(h, g)^{xb} = e(h, g^{xb}) \tag{25}$$

Thus σ^b is a valid signature for the *derived* public key $(g^x)^b$ with blinding value $b \in \mathbb{Z}_q$.

References I

D. Atkins, W. Stallings, and P. Zimmermann.
PGP Message Exchange Formats.
RFC 1991 (Informational), August 1996.
Obsoleted by RFC 4880.

Dan Boneh, Ben Lynn, and Hovav Shacham.
Short signatures from the weil pairing.
In *Advances in Cryptology – ASIACRYPT '01, LNCS*, pages 514–532. Springer, 2001.

J. Callas, L. Donnerhacke, H. Finney, D. Shaw, and R. Thayer.
OpenPGP Message Format.
RFC 4880 (Proposed Standard), November 2007.
Updated by RFC 5581.

References II

J. Callas, L. Donnerhacke, H. Finney, and R. Thayer.
OpenPGP Message Format.
RFC 2440 (Proposed Standard), November 1998.
Obsoleted by RFC 4880.

M. Elkins, D. Del Torto, R. Levien, and T. Roessler.
MIME Security with OpenPGP.
RFC 3156 (Proposed Standard), August 2001.

M. Schanzenbach, C. Grothoff, and B. Fix.
The gnu name system.
RFC 9498, RFC Editor, November 2023.

Elias Summermatter and Christian Grothoff.
Byzantine fault tolerant set reconciliation.
`https://datatracker.ietf.org/doc/html/draft-summermatter-set-union`, 1 2021.

References III

Matthias Wachs, Martin Schanzenbach, and Christian Grothoff.
A censorship-resistant, privacy-enhancing and fully decentralized name system.
In *13th International Conference on Cryptology and Network Security (CANS 2014)*, pages 127–142, 2014.