Decentralized Public Key Infrastructures

Christian Grothoff

Berner Fachhochschule

1.6.2018

Learning Objectives

Learn about:

- Ideas behind the Web of Trust
- Using GnuPG
- Goals and theory behind Fog of Trust
- Semantics of the GNU Name System

GnuPG

- Free version of PGP, with library (libgcrypt)
- Provides common cryptographic primitives
- Provides implementation of OpenPGP (RFC 2440)
- Commonly used for secure E-mail
- Provides web of trust

Using GnuPG

$ gpg –gen-key

$ gpg –export

$ gpg –import FILENAME

$ gpg –edit-key EMAIL; > fpr > sign > trust

$ gpg –clearsign FILENAME

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.

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.

Excercise: Explore

`http://pgp.mit.edu`

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:

$$e : G_1 \times G_2 \to G_T \tag{1}$$

which satisfies $e \neq 1$ and bilinearity:

$$\forall_{a,b \in F_q^*},\ \forall_{P \in G_1, Q \in G_2} :\ e\left(aP, bQ\right) = e\left(P, Q\right)^{ab} \tag{2}$$

Examples: Weil pairing, Tate pairing.

Hardness assumption

Computational Diffie Hellman:

$$g, g^x, g^y \Rightarrow g^{xy} \tag{3}$$

remains hard on G even given e.

Boneh-Lynn-Sacham (BLS) signatures

Key generation:

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

Signing:

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

Verification:

Given public key g^x:

$$e(\sigma, g) = e(h, g^x) \quad (4)$$

Boneh-Lynn-Sacham (BLS) signatures

Key generation:

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

Signing:

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

Verification:

Given public key g^x:

$$e(\sigma, g) = e(h, g^x) \tag{4}$$

Why:

$$e(\sigma, g) = e(h, g)^x = e(h, g^x) \tag{5}$$

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:

$$e(\sigma^b, g) = e(h, g)^{xb} = e(h, g^{xb}) \tag{6}$$

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

Break

The Fog of Trust

Problem:

- Publishing who certified whom exposes the social graph.
- The “NSA kills based on meta data”.

The Fog of Trust

Problem:

- Publishing who certified whom exposes the social graph.
- The "NSA kills based on meta data".

Solution:

- Do not publish the graph.
- Have Alice and Bob collect their certificates locally.
- Use SMC protocol for

  private set intersection cardinality with signatures!

We will only consider paths with **one** intermediary.

Straw-man version of protocol 1

Problem: Alice wants to compute $n := |\mathcal{L}_A \cap \mathcal{L}_B|$

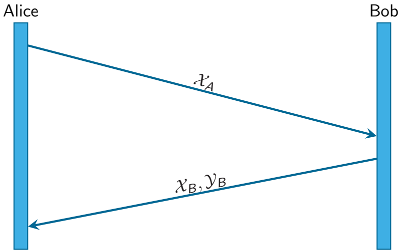
Suppose each user has a private key c_i and the corresponding public key is $C_i := g^{c_i}$ where g is the generator

The setup is as follows:

- $\mathcal{L}_A$: set of public keys representing Alice trusted verifiers
- $\mathcal{L}_B$: set of public keys representing Bob's signers
- Alice picks an ephemeral private scalar $t_A \in \mathbb{F}_p$
- Bob picks an ephemeral private scalar $t_B \in \mathbb{F}_p$

Straw-man version of protocol 1

$\mathcal{X}_A := \{\, C^{t_A} \mid C \in \mathcal{L}_A \,\}$



$$\mathcal{X}_B := \{\, C^{t_B} \mid C \in \mathcal{L}_B \,\}$$
$$\mathcal{Y}_B := \left\{\, \overline{C}^{t_B} \,\middle|\, \overline{C} \in \mathcal{X}_A \,\right\} = \{\, C^{t_B \cdot t_A} \mid C \in \mathcal{L}_B \,\}$$

$$\mathcal{Y}_A := \left\{\, \hat{C}^{t_A} \,\middle|\, \hat{C} \in \mathcal{X}_B \,\right\} = \{\, C^{t_A \cdot t_B} \mid C \in \mathcal{L}_A \,\}$$

Alice can get $|\mathcal{Y}_A \cap \mathcal{Y}_B|$ at linear cost.

Attack against the Straw-man

If Bob controls two trusted verifiers $C_1, C_2 \in \mathcal{L}_A$, he can:

- Detect relationship between $C_1^{t_A}$ and $C_2^{t_A}$
- Choose $K \subset \mathbb{F}_p$ and substitute with fakes:

$$\mathcal{X}_B := \bigcup_{k \in K} \left\{ C_1^k \right\}$$

$$\mathcal{Y}_B := \bigcup_{k \in K} \left\{ (C_1^{t_A})^k \right\}$$

so that Alice computes $n = |K|$.

Cut & choose version of protocol 1: Preliminaries

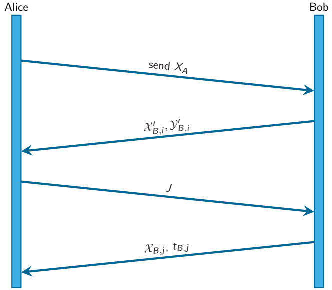
Assume a fixed system security parameter $\kappa \geq 1$.

Let Bob use secrets $t_{B,i}$ for $i \in \{1, \ldots, \kappa\}$, and let $\mathcal{X}_{B,i}$ and $\mathcal{Y}_{B,i}$ be blinded sets over the different $t_{B,i}$ as in the straw-man version.

For any list or set Z, define

$$Z' := \{h(x) | x \in Z\} \tag{7}$$

Cut & choose version of protocol 1



Protocol messages:

1. Alice sends:
   $\mathcal{X}_A :=$
   $\texttt{sort}\,[\,C^{t_A} \mid C \in \mathcal{A}\,]$
2. Bob responds with commitments:
   $\mathcal{X}'_{B,i}, \mathcal{Y}'_{B,i}$ *for* $i \in 1, \ldots, \kappa$
3. Alice picks a non-empty random subset $J \subseteq \{1, \ldots, \kappa\}$ and sends it to Bob.
4. Bob replies with $\mathcal{X}_{B,j}$ for $j \in J$, and $t_{B,j}$ for $j \notin J$.

Cut & choose version of protocol 1: Verification

For $j \notin J$, Alice checks the $t_{B,j}$ matches the commitment $\mathcal{Y}'_{B,j}$.

For $j \in J$, she verifies the commitment to $\mathcal{X}_{B,j}$ and computes:

$$\mathcal{Y}_{A,j} := \left\{ \hat{C}^{t_A} \;\middle|\; \hat{C} \in \mathcal{X}_{B,j} \right\} \tag{8}$$

To get the result, Alice computes:

$$n = |\mathcal{Y}'_{A,j} \cap \mathcal{Y}'_{B,j}| \tag{9}$$

Alice checks that the n values for all $j \in J$ agree.

Protocol 2: Private Set Intersection with Subscriber Signatures

- Naturally, signers are willing to *sign* that Bob's key is Bob's key.
- We still want the identities of the signers to be private!
- BLS (Boneh et. al) signatures are compatible with our blinding.

$\Rightarrow$ Integrate them with our cut & choose version of the protocol.

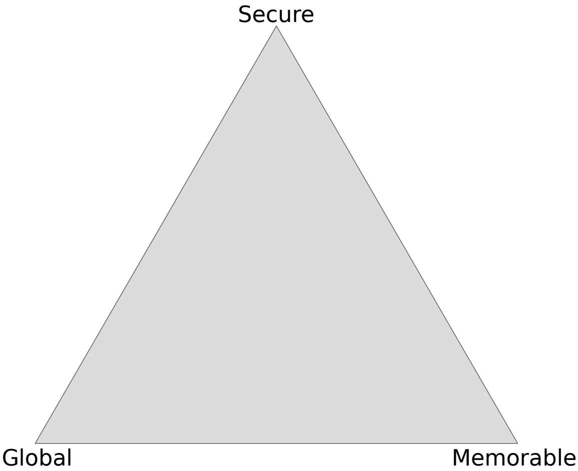
Costs are linear in set size. Unlike prior work this needs no CA.

Break

Security Goals for Name Systems

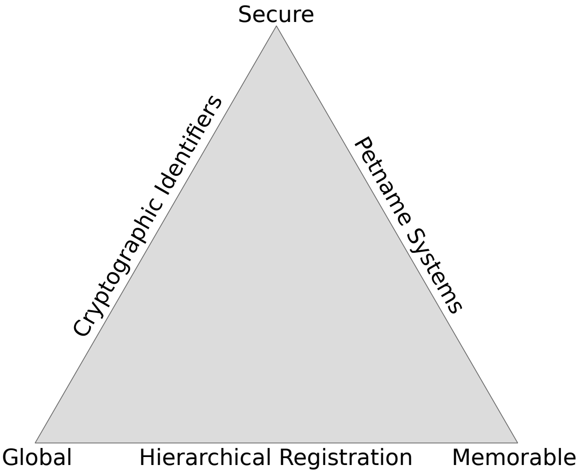
- Query origin anonymity
- Data origin authentication and integrity protection
- Zone confidentiality
- Query and response privacy
- Censorship resistance
- Traffic amplification resistance
- Availability

Zooko's Triangle



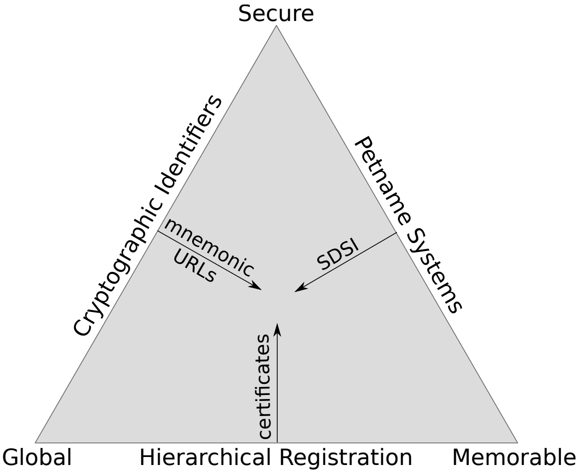
A name system can only fulfill **two**!

Zooko's Triangle

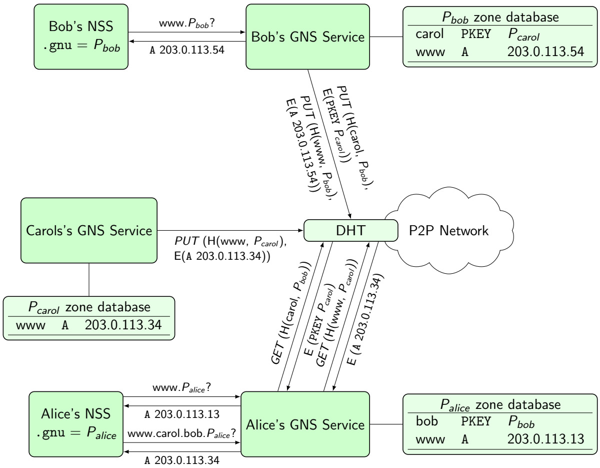


DNS, ".onion" IDs and `/etc/hosts/` are representative designs.

Zooko's Triangle



The GNU Name System (GNS)



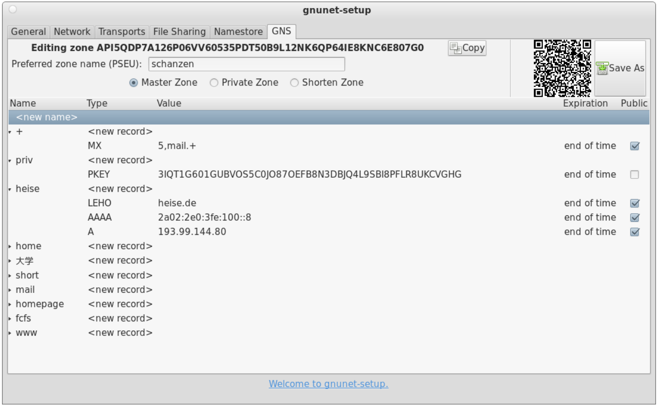
The GNU Name System[1]

Properties of GNS

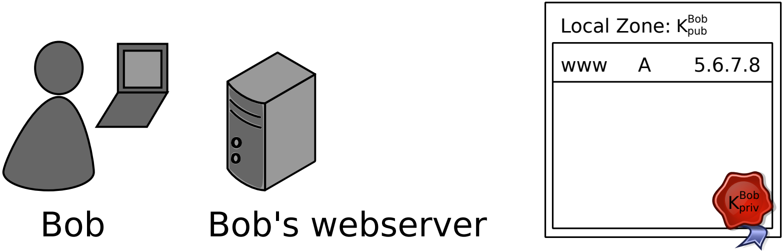
- Decentralized name system with secure memorable names
- Delegation used to achieve transitivity
- Also supports globally unique, secure identifiers
- Achieves query and response privacy
- Provides alternative public key infrastructure
- Interoperable with DNS

[1] Joint work with Martin Schanzenbach and Matthias Wachs

Zone Management: like in DNS



Name resolution in GNS

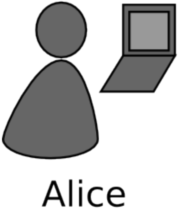


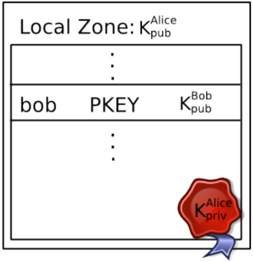
- Bob can locally reach his webserver via **www.gnu**

Secure introduction

- Bob gives his public key to his **friends**, possibly via QR code

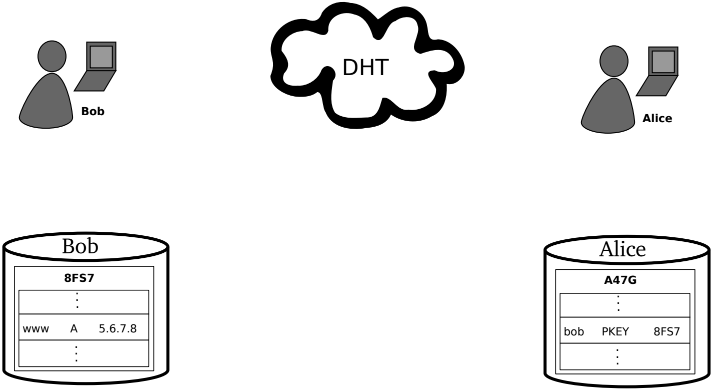
Delegation



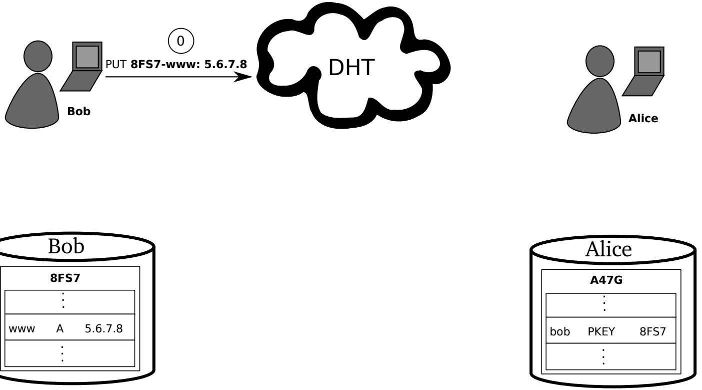


- Alice learns Bob's public key
- Alice creates delegation to zone K_{pub}^{Bob} under label **bob**
- Alice can reach Bob's webserver via **www.bob.gnu**

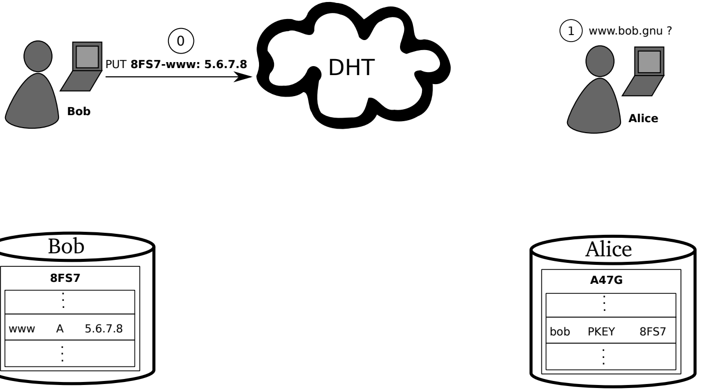
Name Resolution



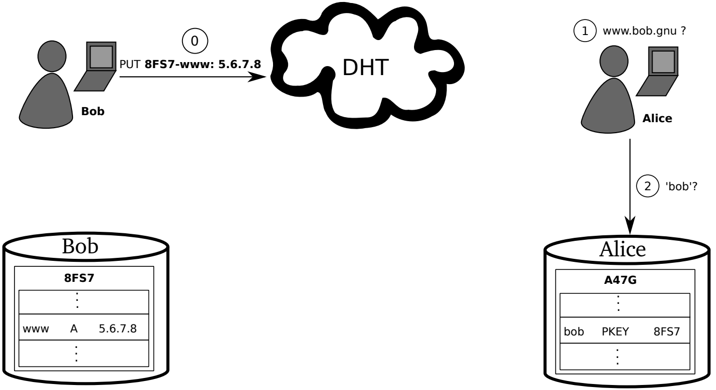
Name Resolution



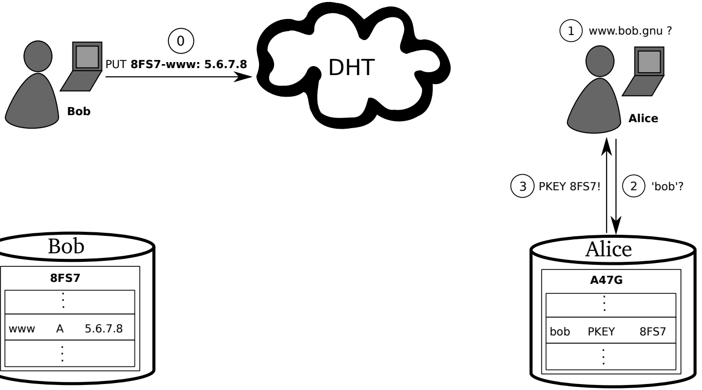
Name Resolution



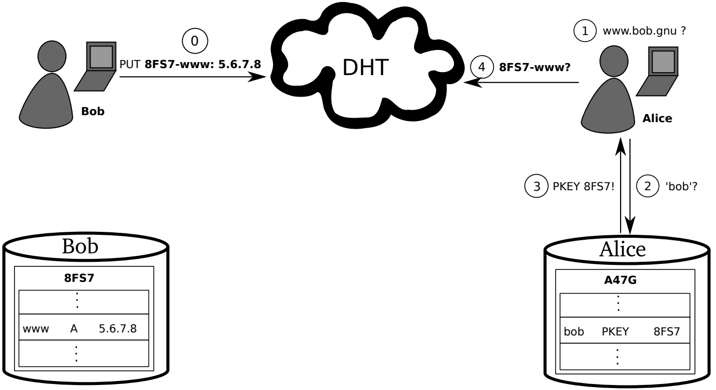
Name Resolution



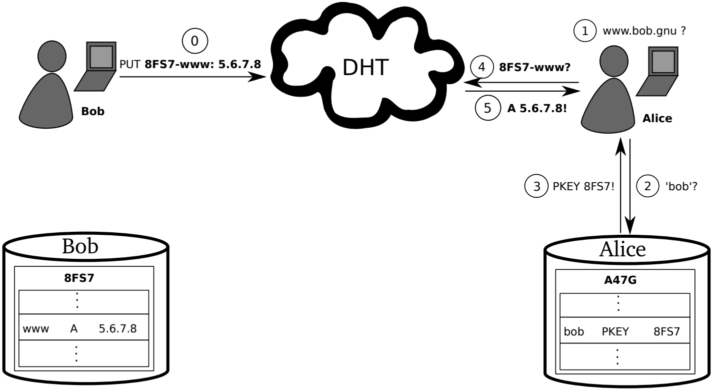
Name Resolution



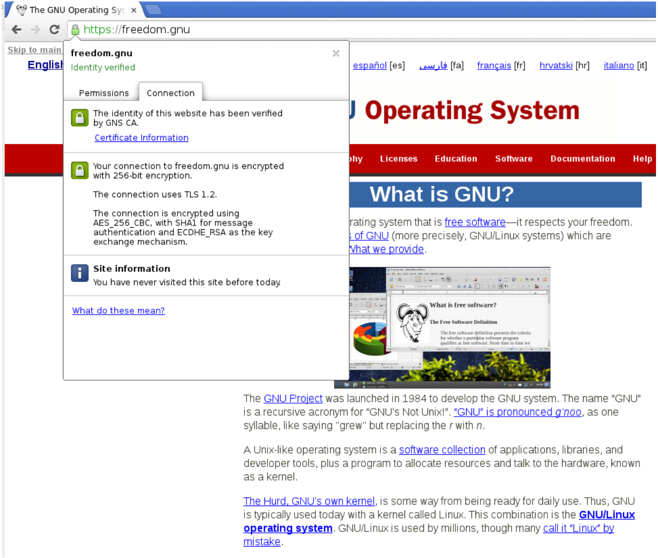
Name Resolution



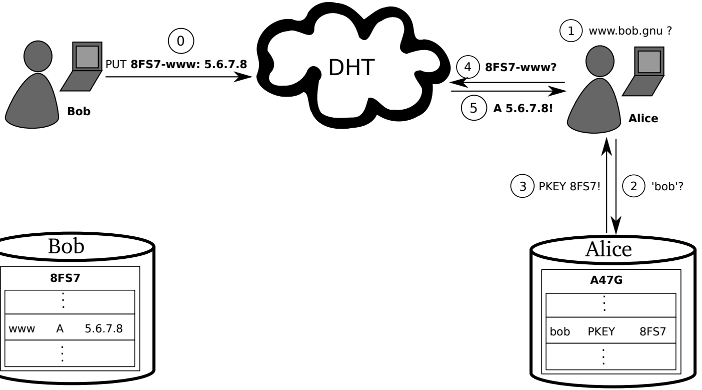
Name Resolution



GNS as PKI (via DANE/TLSA)



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 $q_{P,l}$

$$h := H(l, P) \tag{10}$$

$$d := h \cdot x \mod o \tag{11}$$

$$B_{P,l} := S_d(E_{HKDF(l,P)}(R_{P,l})), dG \tag{12}$$

$$q_{P,l} := H(dG) \tag{13}$$

Query Privacy: Cryptography

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

$$
\begin{aligned}
h &:= H(l, P) && (10)\\
d &:= h \cdot x \mod o && (11)\\
B_{P,l} &:= S_d(E_{HKDF(l,P)}(R_{P,l})), dG && (12)\\
q_{P,l} &:= H(dG) && (13)
\end{aligned}
$$

Searching for records under label l in zone P

$$
\begin{aligned}
h &:= H(l, P) && (14)\\
q_{P,l} &:= H(hP) = H(hxG) = H(dG) \Rightarrow \texttt{obtain } B_{P,l} && (15)\\
R_{P,l} &= D_{HKDF(l,P)}(B_{P,l}) && (16)
\end{aligned}
$$

Using cryptographic identifiers

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

⇒ Globally unique identifiers

Key Revocation

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

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

Alternatives

- DNSSEC
- DNSCurve
- DNS-over-TLS
- Namecoin / Ethereum Name System (ENS)
- RAINS

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	✓	✓	✓	✓	✓	✓	✓

Ongoing and Future Work (Project 2, BS theses)

- Optimze GNUnet DHT
- Implement & evaluate bounded Eppstein set reconciliation
- Integrate GNS with Tor

Conclusion

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

In which world do you want to live?

Exercise

```
# apt-get install git autoconf automake autopoint gettext
# apt-get install libunistring-dev libgnutls28-dev
# apt-get install openssl gnutls-bin libtool libltdl
# apt-get install libcurl-gnutls-dev libidn11-dev
# apt-get install libsqlite3-dev
$ git clone git://gnunet.org/libmicrohttpd
$ git clone git://gnunet.org/gnunet
$ git clone 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-setup  # enable TCP transport only
$ 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
```