RFC 9498: The GNU Name System

Christian Grothoff, Martin Schanzenbach and Bernd Fix

Berner Fachhochschule

22.12.2023
The GNU Name System

- 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
Bob can locally reach his webserver via `www.gns.alt`.

Local Zone:

```
www     A       5.6.7.8
Bob     Bob's webserver
K
Bob
pub
K
Bob
priv
```
Bob Builder, Ph.D.

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

Bob gives his public key to his friends, possibly via QR code.
Delegation

- Alice learns Bob’s public key
- Alice creates delegation to zone $K^{Bob}_{pub}$ under label `bob`
- Alice can reach Bob’s webserver via `www.bob.gns.alt`
Name Resolution

Bob

DHT

Alice

Bob

Alice

Bob

Alice

www      A      5.6.7.8

8FS7

A47G

bob     PKEY       8FS7   

Alice
Name Resolution

Bob

PUT 8FS7-www: 5.6.7.8

DHT

Alice

Bob

<table>
<thead>
<tr>
<th>8FS7</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
</tr>
<tr>
<td>www  A  5.6.7.8</td>
</tr>
<tr>
<td>...</td>
</tr>
</tbody>
</table>

Alice

<table>
<thead>
<tr>
<th>A47G</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
</tr>
<tr>
<td>bob  PKEY  8FS7</td>
</tr>
<tr>
<td>...</td>
</tr>
</tbody>
</table>
Name Resolution

Bob

PUT 8FS7-www: 5.6.7.8

DHT

Alice

1 www.bob.gnu.alt?

Bob

8FS7

www A 5.6.7.8

Alice

A47G

bob PKEY 8FS7
Name Resolution

1. Bob puts `PUT 8FS7-www: 5.6.7.8` into the DHT.
2. Alice queries the DHT for `bob`.
3. The DHT responds with the information stored for `bob`.

Bob's database:

- `8FS7`:
  - `www A 5.6.7.8`

Alice's database:

- `A47G`:
  - `bob PKEY 8FS7`
Name Resolution

1. PUT 8FS7-www: 5.6.7.8
2. 'bob'?
3. PKEY 8FS7!

DHT

Bob

Alice

8FS7

www A 5.6.7.8

A47G

bob PKEY 8FS7
Name Resolution

Bob

DHT

Alice

PUT 8FS7-www: 5.6.7.8

www.bob.gnu.alt?

1

Bob

A47G

www  A  5.6.7.8

Bob

Alice

8FS7

www            A        5.6.7.8

bob PKEY 8FS7

8FS7 www?

PKEY 8FS7!

'bob'?

8FS7-www?

www.bob.gnu.alt?
Name Resolution

1. **PUT** 8FS7-www: 5.6.7.8
2. 'bob'?
3. PKEY 8FS7!
4. 8FS7-www?
5. A 5.6.7.8!

Bob

- 8FS7
- www: A 5.6.7.8

Alice

- A47G
- bob: PKEY 8FS7

DHT

www.bob.gnu.alt?
Privacy Issue: DHT

1. Bob sends `PUT 8FS7-www: 5.6.7.8` to the DHT.
2. Bob asks for the key `bob`.
3. Alice sends her public key `8FS7`.
4. Alice asks for the resource `8FS7-www`.
5. Alice sends the resource `A 5.6.7.8` to Bob.

Bob's local storage:
- 8FS7
- www A 5.6.7.8
- ...

Alice's local storage:
- A47G
- ...
- bob PKEY 8FS7
- ...

www.bob.gnu.alt ?
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 $q_{P,l}$
Publishing records $R_{P,l}$ as $B_{P,l}$ under key $q_{P,l}$

\[
h : = H(l, P) \quad \text{(1)}
\]
\[
d : = h \cdot x \mod o \quad \text{(2)}
\]
\[
B_{P,l} : = S_d(E_{HKDF(l,P)}(R_{P,l})), dG \quad \text{(3)}
\]
\[
q_{P,l} : = H(dG) \quad \text{(4)}
\]
Query Privacy: Cryptography

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

\[ h := H(l, P) \]  
\[ d := h \cdot x \mod o \]  
\[ B_{P,l} := S_d(E_{HKDF(l, P)}(R_{P,l})), dG \]  
\[ q_{P,l} := H(dG) \]

Searching for records under label $l$ in zone $P$

\[ h := H(l, P) \]  
\[ q_{P,l} := H(hP) = H(hxG) = H(dG) \Rightarrow \text{obtain } B_{P,l} \]  
\[ R_{P,l} = D_{HKDF(l, P)}(B_{P,l}) \]
Key Revocation

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

- Certificate Revocation Lists (X.509)
- Online Certificate Status Protocol (OCSP)
- OCSP stapling (TLS)
- Publish revocations in a 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
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:

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

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

\( b = \text{Contains}(d, e) \) Check if BF \( d \) contains element \( e \).

\( b \in \{ \text{“Definitely not in set”}, \text{“Probably in set”} \} \)
BF: Insert

Element #1 → H

\[ H(\text{Element \#1}) = (2, 3, 7) \]
BF: Insert

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

Element #2 → H

\[ H(\text{Element } \#1) = (2, 3, 7) \]
\[ H(\text{Element } \#2) = (1, 3, 5) \]
BF: Insert

Element #2 \rightarrow H

\begin{align*}
H(\text{Element } #1) &= (2, 3, 7) \\
H(\text{Element } #2) &= (1, 3, 5)
\end{align*}
BF: Membership Test

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

$H(\text{Element } \#2) = (1, 3, 5)$
BF: Membership Test (false positive)

Element #4

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

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

BF where buckets hold a positive integer.

Additional Operation:

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

⇒ 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:

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

\[d' = \text{SymDiff}(d_1, d_2)\] Create an IBF that represents the symmetric difference of \(d_1\) and \(d_2\).
IBF: Insert Element #1

$H$(Element #1) $\mapsto$ (2, 3, 7)
$H'(Element$ #1) $\mapsto$ 4242 (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

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

$H'(\text{Element } \#2) = 0101$ (ID)

$H''(0101) \to 41$
### IBF: Insert Element #2

<table>
<thead>
<tr>
<th>H</th>
<th>0101</th>
<th>41</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4242</td>
<td>13</td>
</tr>
<tr>
<td>2</td>
<td>4343</td>
<td>52</td>
</tr>
<tr>
<td>0</td>
<td>0000</td>
<td>00</td>
</tr>
</tbody>
</table>

\[ H(\text{Element \#2}) = (1, 3, 5) \]

\[ H'(\text{Element \#2}) = 0101 \text{ (ID)} \]

\[ H''(0101) \mapsto 41 \]
Symmetric Difference on IBFs

We can directly compute the symmetric difference without extraction.

- Subtract counters
- XOR of IDSUM and XHASH values
<table>
<thead>
<tr>
<th>IBF: Extract</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Table and Diagram" /></td>
</tr>
</tbody>
</table>

- **Impure bucket** ⇒ potential decoding failure
- **Pure bucket** ⇒ extractable element ID
- **Extraction** ⇒ more pure buckets (hopefully/probably)
- **Less elements** ⇒ more chance for pure buckets

- $|\text{counter}| = 1 \land H''(\text{IDSUM}) = X\text{HASH} \leftrightarrow \text{pure}$
The Set Union Protocol

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

- Amount of communication and computation only depends on $\delta$, 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 $SE_{diff} = \text{SymDiff}(SE_{Alice}, SE_{Bob})$
   - by pair-wise $\text{SymDiff}$ of all IBFs in the SE

4. Alice estimates the size of $SE_{diff}$. 
Strata Estimator

IBF 1

IBF 2

IBF 3

IBF 4
Strata Estimator
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 $\delta$
3. Bob computes $IBF_{Bob}$ for size $\delta$ and sends it to Alice
4. Alice computes $IBF_{Alice}$
5. Alice computes $IBF_{diff} = SymDiff(\text{IBF}_{Alice}, \text{IBF}_{Bob})$
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!
## Privacy summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Defense against MiTM</th>
<th>Zone privacy</th>
<th>Privacy vs. network</th>
<th>Privacy vs. operator</th>
<th>Traffic amplification resistance</th>
<th>Censorship resistance</th>
<th>Ease of migration</th>
</tr>
</thead>
<tbody>
<tr>
<td>DNS</td>
<td>✗</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>DNSSEC</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>DNSCurve</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>DNS-over-TLS</td>
<td>✓</td>
<td>n/a</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Namecoin</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>RAINS</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>GNS</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
</tbody>
</table>

*EDNS0
# Key management summary

<table>
<thead>
<tr>
<th></th>
<th>Suitable for personal use</th>
<th>Memorable</th>
<th>Decentralised</th>
<th>Modern cryptography</th>
<th>Understandable</th>
<th>Exposes metadata</th>
<th>Transitive</th>
</tr>
</thead>
<tbody>
<tr>
<td>DNS</td>
<td>✗</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>DNSSEC</td>
<td>✗</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>DNSCurve</td>
<td>✗</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>DNS-over-TLS</td>
<td>✗</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>TLS-X.509</td>
<td>✗</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>Web of Trust</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>TOFU</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Namecoin</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>RAINS</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>GNS</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>
Conclusion

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>DNS</td>
<td>globalist</td>
</tr>
<tr>
<td>DNSSEC</td>
<td>authoritarian</td>
</tr>
<tr>
<td>Namecoin</td>
<td>libertarian</td>
</tr>
<tr>
<td>RAINS</td>
<td>nationalist</td>
</tr>
<tr>
<td>GNS</td>
<td>anarchist</td>
</tr>
</tbody>
</table>

In which world do you want to live?