Decentralized Public Key Infrastructures

Christian Grothoff

Berner Fachhochschule

24.5.2019
Learning Objectives

Learn about:
- Blockchain Blockchain Blockchain!
- Ideas behind the Web of Trust
- Using GnuPG
- Goals and theory behind Fog of Trust
- Modern public key infrastructures
Blockchain

1Illustrations by Alexandra Dirksen, IAS, TUBS [1]
Blockchain

Charlie Peter
Blockchain

Charlie Peter
Blockchain
Blockchain

1. Bob, Alice

2. Charlie, Peter
Blockchain

1. Bob Alice

2. Charlie Peter

3. Eve Alex
Blockchain
Blockchain
Advertised Blockchain “properties”
Immutability
Transparency
Decentralisation
Anonymity
Blockchain “properties”

- Immutability
- Transparency
- Anonymity
- Decentralisation
- Irreversibility
- Autonomy

These only hold with many significant caveats!
Who gets to append the next block?
Proof of Work
Proof of Work
Proof of Work
Proof of Work
Proof of Work
Proof of Work
Break
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.

Solution:
- Find paths in the certification graph from Alice to Bob.
- If sufficient number of short paths exist certifying the same key, trust it.
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 $e$:

$$e : G_1 \times G_2 \rightarrow G_T$$

which satisfies $e \neq 1$ and bilinearity:

$$\forall a, b \in \mathbb{F}_q^*, \forall P \in G_1, Q \in G_2 : e(aP, bQ) = e(P, Q)^{ab}$$

Examples: Weil pairing, Tate pairing.
Hardness assumption

Computational Diffie Hellman:

\[ g, g^x, g^y \Rightarrow g^{xy} \]  \hspace{2cm} (3)

remains hard on \( G \) even given \( e \).
Boneh-Lynn-Saham (BLS) signatures

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

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

Verification:
Given public key $g^x$:

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

(4)
Boneh-Lynn-Sacham (BLS) signatures

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

Signing:
\[ \sigma := h^x \text{ where } h := H(m) \]

Verification:
Given public key $g^x$:
\[ e(\sigma, g) = e(h, g^x) \] (4)

Why:
\[ e(\sigma, g) = e(h, g)^x = e(h, g^x) \] (5)
due to bilinearity.
Given signature $\langle \sigma, g^x \rangle$ on message $h$, we can *blind* the signature and public key $g^x$:

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

(6)

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

Fun with BLS
Break
The Fog of Trust

**Problem:**

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

\[ \mathcal{X}_A : = \{ C^{t_A} \mid C \in \mathcal{L}_A \} \]

\[ \mathcal{Y}_A : = \{ \hat{C}^{t_A} \mid \hat{C} \in \mathcal{X}_B \} \]
\[ = \{ C^{t_A \cdot t_B} \mid C \in \mathcal{L}_B \} \]

\[ \mathcal{X}_B : = \{ C^{t_B} \mid C \in \mathcal{L}_B \} \]
\[ \mathcal{Y}_B : = \{ \overline{C}^{t_B} \mid \overline{C} \in \mathcal{X}_A \} \]
\[ = \{ C^{t_B \cdot t_A} \mid C \in \mathcal{L}_B \} \]

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:

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

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

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

(7)
Cut & choose version of protocol 1

Protocol messages:

1. Alice sends:
   \[ \mathcal{X}_A := \text{sort} \left[ C^t_A \mid C \in \mathcal{A} \right] \]
2. Bob responds with commitments:
   \[ \mathcal{X}'_{B,i}, \mathcal{Y}'_{B,i} \quad \text{for} \quad i \in 1, \ldots, \kappa \]
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:

\[
\mathcal{Y}_{A,j} := \left\{ \hat{C}^{t_A} \mid \hat{C} \in \mathcal{X}_{B,j} \right\}
\] (8)

To get the result, Alice computes:

\[
n = |\mathcal{Y}_{A,j}' \cap \mathcal{Y}_{B,j}'|
\] (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
A name system can only fulfill two!
Zooko’s Triangle

Secure

Global

Hierarchical Registration

Memorable

Cryptographic Identifiers

Petname Systems

DNS, “.onion” IDs and /etc/hosts/ are representative designs.
Zooko’s Triangle

- Secure
- Cryptographic Identifiers
- mnemonic
- URLs
- certificates
- SDSI
- Petname Systems
- Hierarchical Registration
- Global
- Memorable
Approaches Adding Cryptography to DNS

- DNSSEC
- DNSCurve
- DNS-over-TLS
- DNS-over-HTTPS
- RAINS
Case study: DoH

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 IETF is standardizing DNS over HTTPS (DOH), where all DNS queries are sent over the HTTPS protocol to some well-known HTTPS server (such as Google’s 8.8.8.8 or Cloudflare’s 1.1.1.1). This will prevent local governments from manipulating DNS traffic and improve the user’s privacy with respect to their ISPs and governments. However, Google or Cloudflare will see the DNS queries and replies of the users, and they must be expected to have weak privacy policies and are subject to US law which includes secret rules and court orders. The NSA has a history of snooping on (MORECOWBELL) and manipulating (QUANTUMDNS) DNS traffic.”

Discuss virtues and vices affected.
Case study: RAINS

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 ETH Zurich is developing a new name system called RAINS with a new trust anchor operated by the regional Internet service provides, aka the local Isolation Service Domain (ISD). RAINS does not change the privacy of DNS (provides can continue to monitor traffic, all zone data becomes public) and allows the local authorities to block Web sites to improve public safety and enforce local laws (see also: "Glücksspielgesetz in Switzerland"). At the same time, foreign censorship efforts are less likely to be effective (unless they foreign government forces the DNS authority to alter the authoritative records”).

Discuss virtues and vices affected.
Break
The GNU Name System (GNS) [2]

Bob's NSS
\[ .gnu = P_{bob} \]

Alice's NSS
\[ .gnu = P_{alice} \]

Carol's NSS
\[ .gnu = P_{carol} \]

Bob's GNS Service
\[ \text{www.}P_{bob}? \]

Alice's GNS Service
\[ \text{www.}P_{alice}? \]

P2P Network

DHT

PUT (H(www, P_{carol}), E(A 203.0.113.34))
GET (H(www, P_{carol}), E(A 203.0.113.34))
The GNU Name System

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

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 Builder, Ph.D.
Address: Country, Street Name 23
Phone: 555-12345
Mobile: 666-54321
Mail: bob@H2R84L4JIL3G5C.zkey

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

Bob

Alice

DHT

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

<table>
<thead>
<tr>
<th>Alice</th>
<th>A47G</th>
</tr>
</thead>
<tbody>
<tr>
<td>bob</td>
<td>PKEY</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Name Resolution

Bob

PUT 8FS7-www: 5.6.7.8

DHT

Alice

...www      A      5.6.7.8
8FS7

Bob

A47G
...
...
bob     PKEY       8FS7
Alice

...www      A      5.6.7.8
Name Resolution

Bob

PUT 8FS7-www: 5.6.7.8

DHT

Alice

www.bob.gnu ?

Bob

Alice

8FS7

www A 5.6.7.8

A47G

bob PKEY 8FS7
Name Resolution

1. www.bob.gnu

2. 'bob'?

Bob

DHT

Alice

0. PUT 8FS7-www: 5.6.7.8

1. www.bob.gnu ?

2. 'bob'?

Bob

8FS7

www A 5.6.7.8

Alice

A47G

bob PKEY 8FS7
Name Resolution

Bob

PUT 8FS7-www: 5.6.7.8

DHT

Alice

1 www.bob.gnu ?

2 'bob'? PKEY 8FS7!

3 PKEY 8FS7!

Bob

8FS7

... 

www A 5.6.7.8

...

Alice

A47G

... 

bob PKEY 8FS7

...
Name Resolution

Bob

PUT 8FS7-www: 5.6.7.8

DHT

Alice

www.bob.gnu?

1

2 'bob'?

3 PKEY 8FS7!

0

4 8FS7-www?

www      A      5.6.7.8 

8FS7

Bob

8FS7

www A 5.6.7.8

Alice

A47G

bob PKEY 8FS7
Name Resolution

0. PUT 8FS7-www: 5.6.7.8

1. www.bob.gnu ?

2. 'bob'? PKEY 8FS7!

3. PKEY 8FS7!

4. 8FS7-www?

5. A 5.6.7.8!

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>
GNS as PKI (via DANE/TLSA)

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 “gnew”, 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 as a 'theory of operating system'.
Privacy Issue: DHT

Bob

PUT 8FS7-www: 5.6.7.8

DHT

Alice

1. www.bob.gnu?

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 ...

...
Query Privacy: Terminology

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

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

\begin{align*}
  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)
\end{align*}
Query Privacy: Cryptography

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

\[
h : = H(l, P) \quad \text{(10)}
\]
\[
d : = h \cdot x \mod o \quad \text{(11)}
\]
\[
B_{P,l} : = S_d(E_{HKDF(l,P)}(R_{P,l})) , dG \quad \text{(12)}
\]
\[
q_{P,l} : = H(dG) \quad \text{(13)}
\]

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

\[
h : = H(l, P) \quad \text{(14)}
\]
\[
q_{P,l} : = H(hP) = H(hxG) = H(dG) \Rightarrow \text{obtain } B_{P,l} \quad \text{(15)}
\]
\[
R_{P,l} = D_{HKDF(l,P)}(B_{P,l}) \quad \text{(16)}
\]
Using cryptographic identifiers

- Zone are identified by a public key
- “alice.bob.\textit{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
Efficient Set Union
(based on “What’s the difference? Efficient Set Reconciliation without Prior Context”, Eppstein et al., SIGCOMM’11)

- 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
Bad 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.

\[ b \in \{ \text{“Definitely not in set”}, \text{“Probably in set”} \} \]
Element #1 \rightarrow H

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

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

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

Element #2 → H

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

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

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

Element #4 \rightarrow H

H(Element #1) = (2, 3, 7)
H(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**
- Additionally store **(XOR-)sum of hashes** in buckets.

Additional Operations:

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

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

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

$H'(\text{Element \#1}) = 4242$
IBF: Insert

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

\[
H'(\text{Element } \#1) = 4242
\]
IBF: Insert

\( H(\text{Element } \#1) = (2, 3, 7) \)
\( H'(\text{Element } \#1) = 4242 \)
\( H(\text{Element } \#2) = (1, 3, 5) \)
\( H'(\text{Element } \#2) = 0101 \)
**IBF: Insert**

<table>
<thead>
<tr>
<th>Element</th>
<th>H</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0000</td>
</tr>
<tr>
<td>1</td>
<td>0101</td>
</tr>
<tr>
<td>1</td>
<td>4242</td>
</tr>
<tr>
<td>2</td>
<td>4343</td>
</tr>
<tr>
<td>0</td>
<td>0000</td>
</tr>
<tr>
<td>1</td>
<td>0101</td>
</tr>
<tr>
<td>1</td>
<td>4242</td>
</tr>
</tbody>
</table>

\[
H(\text{Element } \#1) = (2, 3, 7) \\
H'(\text{Element } \#1) = 4242 \\
H(\text{Element } \#2) = (1, 3, 5) \\
H'(\text{Element } \#2) = 0101
\]
IBF: Extract

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>2</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>2</td>
<td>4</td>
<td>2</td>
</tr>
</tbody>
</table>

- Pure bucket ⇒ extractable element hash
- Extraction ⇒ more pure buckets (hopefully/probably)
- Less elements ⇒ more chance for pure buckets
Symmetric Difference

We can directly compute the symmetric difference without extraction.

- Subtract counts
- XOR hashes
The Set Union Protocol

1. ⇒ Create IBFs
2. Compute SymDiff
3. Extract element hashes

- Amount of communication and computation only depends on $\delta$, not $|A| + |B|$.
- How do we choose the initial size of the IBF?
- ⇒ 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 0 contains $1/2$ of all elements
   - Stratum 1 contains $1/4$ of all elements
   - Stratum $n$ contains $1/(2^n)$ all elements

2. Alice receives Bob’s strata estimator

3. Alice computes $SE_{diff} = SymDiff(SE_{Alice}, SE_{Bob})$
   - by pair-wise $SymDiff$ of all IBFs in the SE

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

- IBF 3
- IBF 2
- IBF 1
- IBF 0
Strata Estimator

IBF 3

IBF 2

IBF 1

IBF 0

3

7
Strata Estimator

IBF 0

IBF 1

IBF 2

IBF 3

??

7

3
Estimation

Estimate as $(3 + 7) \cdot 2^4$. (Number of extracted hashes scaled by expected number of elements in the remaining IBFs)
The Complete Protocol

1. Alice sends $SE_{Alice}$ to Bob
2. Bob estimates the set difference $\delta$
3. Bob computes $IBF_{Bob}$ with 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 hashes 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 . . .
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
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.
Break
Let’s just put the records into the Blockchain!
Let’s just put the records into the Blockchain!

Or rather, put the public key of the owner and signed updates into it.
Let’s just put the records into the Blockchain!

Or rather, put the public key of the owner and signed updates into it.

And let’s have some expiration rules.
Case study: Namecoin

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.

“Namecoin establishes a new name system on the blockchain (where thus zone data is also public), but where public authorities cannot block information. Queries are performed against a local copy of the blockchain and thus also private. There is no WHOIS, so the owner of a name can also be anonymous. However, Namecoin uses much more bandwidth and energy as blockchain payments are used for registration and name resolution. Names are registered on a first-come, first-served basis. Trademarks, copyrights anti-fraud or anti-terrorism judgements cannot be used to force owners of names to relinquish names.”

Discuss virtues and vices affected.
Break
Ethereum Name System

Let’s have a smart contract in the Blockchain manage naming!

Blockchain contains smart contract and data who controls which name.

Contract allocates names under .eth using auctions.

https://ens.domains/
Ethereum Name System\textsuperscript{5}

\textsuperscript{5}https://ens.domains/
### Privacy summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Defense against MiTM</th>
<th>Zone privacy</th>
<th>Privacy vs. network</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>
</tr>
<tr>
<td>DNSSEC</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>
</tr>
<tr>
<td>DNS-over-TLS</td>
<td>✓</td>
<td>n/a</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>
</tr>
<tr>
<td>RAINS</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>
</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>

**Key to symbols:**
- ✓: Yes
- ×: No
- ☢: Partially valid
Possible Future Work (Project 2, BS thesis)

- Implement & evaluate **bounded** Eppstein set reconciliation
- Implement Fog-of-Trust (ideally in Rust)
## Conclusion

<table>
<thead>
<tr>
<th>DNS</th>
<th>globalist</th>
</tr>
</thead>
<tbody>
<tr>
<td>DNSSEC</td>
<td>authoritarian</td>
</tr>
<tr>
<td>Namecoin</td>
<td>libertarian (US)</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?
# 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
References

Alexandra Dirksen. 
A blockchain picture book. 

Matthias Wachs, Martin Schanzenbach, and Christian Grothoff. 
A censorship-resistant, privacy-enhancing and fully decentralized name system. 