Riposte: an Anonymous Messaging System that 'Hides the Metadata'

Henry Corrigan-Gibbs

Joint work with Dan Boneh and David Mazières
To appear at IEEE Security and Privacy 2015

Charles River Crypto Day
20 February 2015
With PKE, we can hide the data...

...but does that hide enough?

\[(pk, sk)\]
<table>
<thead>
<tr>
<th>Time</th>
<th>From</th>
<th>To</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>10:12</td>
<td>Alice</td>
<td>Bob</td>
<td>2543 B</td>
</tr>
<tr>
<td>10:27</td>
<td>Carol</td>
<td>Alice</td>
<td>567 B</td>
</tr>
<tr>
<td>10:32</td>
<td>Alice</td>
<td>Bob</td>
<td>450 B</td>
</tr>
<tr>
<td>10:35</td>
<td>Bob</td>
<td>Alice</td>
<td>9382 B</td>
</tr>
<tr>
<td>Time</td>
<td>From</td>
<td>To</td>
<td>Size</td>
</tr>
<tr>
<td>-------</td>
<td>-------</td>
<td>-------------------</td>
<td>-------</td>
</tr>
<tr>
<td>10:12</td>
<td>Alice</td>
<td><a href="mailto:taxfraud@stanford.edu">taxfraud@stanford.edu</a></td>
<td>2543 B</td>
</tr>
<tr>
<td>10:27</td>
<td>Carol</td>
<td>Alice</td>
<td>567 B</td>
</tr>
<tr>
<td>10:32</td>
<td>Alice</td>
<td>Bob</td>
<td>450 B</td>
</tr>
<tr>
<td>10:35</td>
<td>Bob</td>
<td>Alice</td>
<td>9382 B</td>
</tr>
</tbody>
</table>

Hiding the data is necessary, but not sufficient

[cf. Ed Felten’s testimony before the House Judiciary Committee, 2 Oct 2013]
Focus of this talk

**Goal**: post “anonymously” to a public bulletin board

Building block for many problems related to “hiding the metadata”

- E-voting
- Anonymous surveys
- Private messaging, etc.
"Onion" encryption

[Dingledine, Mathewson, Syverson 2004]
Passive network adversary can correlate flows!

Is this attack realistic?

[Murdoch and Danezis 2005]
[Bauer et al. 2007]
A well-placed adversary need control few links

[Murdoch and Zieliński 2007]
Tor is **practical** at Internet scale

… but its security properties are unclear

We design an **anonymous messaging system** that:

1) satisfies clear security goals,

2) handles millions of users in an “anonymous Twitter” system.
Outline

• Motivation
• Definitions and a “Straw man” scheme
• Technical challenges
• Evaluation
• Conclusions
Outline

• Motivation
• Definitions and a “Straw man” scheme
• Technical challenges
• Evaluation
• Conclusions
Reframe the Problem

Posting anonymously to a bulletin board

≡

Writing “privately” to a database
Goal

The “Anonymity Set”
Goal
DB does not learn who wrote which message

To: taxfraud@stanford.edu

Protest will be held tomo…
See my cat photos at w…
(k,t)-Write-Anonymous DB Scheme

\( k = (\text{# of servers}), \ t = (\text{# malicious servers}) \)

\((q_1, \ldots, q_k) \leftarrow \text{Query}(m)\)

- [Gen query to write \( m \) into row \( l \) of DB]

\( s' \leftarrow \text{Update}(s, q_i) \)

- [Apply query to state of server \( i \)]

\( DB \leftarrow \text{Reveal}(s_1, \ldots, s_k) \)

- [Combine server states to reveal plaintext DB]
Goals

1. Correctness
2. Write-anonymity
3. Disruption resistance
Goal 1: Correctness

**Informal:** Output DB should be result of applying queries to DB state.

If queries are: \((m_1, \ell_1), \ldots, (m_n, \ell_n)\) then result of \text{Reveal}() is:

\[
\sum_{i=1}^{n} m_i \in \ell_i
\]
Goal 2: Write-Anonymity

**Informal**: coalition of $t$ malicious servers and any number of malicious clients should not learn who wrote what to the DB.

I will present the [simplified] two-server definition with one malicious server.
Let $n =$ number of clients total

For $i \in H$

$(q_i, \hat{q}_i) \leftarrow \text{Query}(m_i, l_i)$

Choose on $\pi$ elements of $H$

\[
H \subseteq [n] \text{ s.t. } |H| \geq 2
\]

\[
\{ (m_i, l_i) \mid i \in H \}
\]

\[
\{ q_{\pi(i)} \mid i \in H \}
\]
Let \( n \) = number of clients total

\[
H \subseteq [n] \text{ s.t. } |H| \geq 2
\]

\[
\{(m_i, l_i) \mid i \in H\}
\]

\[
\{q_{\pi(i)} \mid i \in H\}
\]

\[
\{\hat{q}_i \mid i \notin H\}
\]
Let $H$ – number of clients total

$$H \subseteq [n] \text{ s.t. } |H| \geq 2$$

$$\{(m_i, \ell_i) \mid i \in H\}$$

For $i \in H$

$$(q_i, \hat{q}_i) \leftarrow \text{Query}(m_i, \ell_i)$$

Choose perm $\pi$ on elements of $H$

$$\{q_{\pi(i)} \mid i \in H\}$$

$$\{\hat{q}_i \mid i \not\in H\}$$

$s \leftarrow \text{Update}$
For \( i \in H \):

\((q_i, \hat{q}_i) \leftarrow \text{Query}(m_i, l_i)\)

Choose perm \( \pi \) on elements of \( H \)

\( s \leftarrow \text{Update} (\hat{q}_1, \ldots, \hat{q}_n) \)
For \( i \in H \):

\((q_i, \hat{q}_i) \leftarrow \) Query\((m_i, \ell_i)\)

Choose \( s \leftarrow \) \text{Update} \((\hat{q}_1, \ldots, \hat{q}_n)\)

Queries in \( H \) updated according to permutation \( \pi \)

\( \{(m_i, \ell_i) \mid i \in H\} \)

\( \{\hat{q}_i \mid i \notin H\} \)

\( S, \pi \)
For $i \in H$:

$(q_i, \hat{q}_i) \leftarrow \text{Query}(m_i, \ell_i)$

Choose perm $\pi$ on elements of $H$

Choose perm $\pi^*$ on elements of $H$

$s \leftarrow \text{Update}:(\hat{q}_1, \ldots, \hat{q}_n)$

\{ $(m_i, \ell_i) \mid i \in H$ \}

\{ $q_{\pi(i)} \mid i \in H$ \}

\{ $\hat{q}_i \mid i \notin H$ \}

\[ S, \pi^* \]

**Intuition**: The scheme hides “who wrote what” (which query corresponds to which message)
Goal 3: Disruption Resistance

**Intuition**: each query should change at most one DB row—prevent disruption

**Informal**: An adversary cannot generate N “valid” queries that affect > N rows

[We defer the definition a “valid” query for now...]
Privacy-Preserving DB Schemes

**ORAM** [GO’96] / Group ORAM [GOMT’11]
– CPU(s) writing to RAM

**Private Info Retrieval (PIR)** [CGKS’97]
– Client reading from DB shared across servers

**Private Info Storage** [OS’97]
– Client writing to DB shared across servers

This work: Many clients (incl malicious ones) writing to DB shared across servers

Ideally: for all $k$, tolerate compromise of $k-1$ servers
(2,1)-Private "Straw man" Scheme

[Chaum '88]
<table>
<thead>
<tr>
<th>$S_X$</th>
<th>$S_Y$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

“Straw man” Scheme
Write msg $m_A$ into DB row 3

$m_A \in \mathbb{F}$

"Straw man" Scheme
"Straw man" Scheme
“Straw man” Scheme
The diagram illustrates the "Straw man" Scheme with the following components:

- Two matrices, $S_X$ and $S_Y$, each with five rows and a single column.
- A third matrix with five elements, $m_A$, $r_1$, $r_2$, $r_3$, and $r_4$, and another with $-r_1$, $-r_2$, $m_A - r_3$, $-r_4$, and $-r_5$.

The equation shown is:

$$S_X - S_Y = \text{"raw man" Scheme}$$
"Straw man" Scheme
"Straw man" Scheme
<table>
<thead>
<tr>
<th>$r_1$</th>
<th>$-r_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$r_2$</td>
<td>$-r_2$</td>
</tr>
<tr>
<td>$r_3$</td>
<td>$-r_3 + m_A$</td>
</tr>
<tr>
<td>$r_4$</td>
<td>$-r_4$</td>
</tr>
<tr>
<td>$r_5$</td>
<td>$-r_5$</td>
</tr>
</tbody>
</table>

“Straw man” Scheme
"Straw man" Scheme
Straw man

Scheme

$$S_X$$

$$r_1$$
$$r_2$$
$$r_3$$
$$r_4$$
$$r_5$$

$$S_Y$$

$$-r_1$$
$$-r_2$$
$$-r_3 + m_A$$
$$-r_4$$
$$-r_5$$

raw man”

Scheme

$$m_B$$

$$s_1$$
$$s_2$$
$$s_3$$
$$s_4$$
$$s_5$$

$$-s_1$$
$$-s_2$$
$$-s_3$$
$$-s_4$$

$$m_B - s_5$$
Straw man

Scheme

\[ S_X \]

\[
\begin{array}{c}
 r_1 \\
 r_2 \\
 r_3 \\
 r_4 \\
 r_5 \\
\end{array}
\]

\[ S_Y \]

\[
\begin{array}{c}
 -r_1 \\
 -r_2 \\
 -r_3 + m_A \\
 -r_4 \\
 -r_5 \\
\end{array}
\]
"Straw man" Scheme
\[ S_X \]

\[
\begin{align*}
  r_1 + s_1 \\
  r_2 + s_2 \\
  r_3 + s_3 \\
  r_4 + s_4 \\
  r_5 + s_5 
\end{align*}
\]

\[ S_Y \]

\[
\begin{align*}
  -r_1 - s_1 \\
  -r_2 - s_2 \\
  -r_3 - s_3 + m_A \\
  -r_4 - s_4 \\
  -r_5 - s_5 - m_B 
\end{align*}
\]

“Straw man” Scheme
<table>
<thead>
<tr>
<th>$S_X$</th>
<th>$S_Y$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$r_1 + s_1$</td>
<td>(-r_1 - s_1)</td>
</tr>
<tr>
<td>$r_2 + s_2$</td>
<td>(-r_2 - s_2)</td>
</tr>
<tr>
<td>$r_3 + s_3$</td>
<td>(-r_3 - s_3 + m_A)</td>
</tr>
<tr>
<td>$r_4 + s_4$</td>
<td>(-r_4 - s_4)</td>
</tr>
<tr>
<td>$r_5 + s_5$</td>
<td>(-r_5 - s_5 - m_B)</td>
</tr>
</tbody>
</table>

“Straw man” Scheme
<table>
<thead>
<tr>
<th>$r_1 + s_1$</th>
<th>$-r_1 - s_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$r_2 + s_2$</td>
<td>$-r_2 - s_2$</td>
</tr>
<tr>
<td>$r_3 + s_3$</td>
<td>$-r_3 - s_3 + m_A$</td>
</tr>
<tr>
<td>$r_4 + s_4$</td>
<td>$-r_4 - s_4$</td>
</tr>
<tr>
<td>$r_5 + s_5$</td>
<td>$-r_5 - s_5 - m_B$</td>
</tr>
</tbody>
</table>

“Straw man” Scheme
\[ S_X \]

- \( r_1 + s_1 \)
- \( r_2 + s_2 \)
- \( r_3 + s_3 \)
- \( r_4 + s_4 \)
- \( r_5 + s_5 \)

\[ S_Y \]

- \(-r_1 - s_1\)
- \(-r_2 - s_2\)
- \(-r_3 - s_3 + m_A\)
- \(-r_4 - s_4\)
- \(-r_5 - s_5 - m_B\)

“Straw man” Scheme
At the end of the day, servers combine DBs to reveal plaintext

**“Straw man” Scheme**
First-Attempt Scheme: Properties

Correctness
— By construction

Write-Anonymity
— Given output vector, servers can simulate their view of the protocol run

Practical Efficiency
— Almost no “heavy” computation involved
Extensions

Use $k > 2$ servers
$\rightarrow$ secure against $k-1$ evil servers

Use a large-characteristic field
$\rightarrow$ e.g., email-length rows
Outline

• Motivation
• Definitions and a “Straw man” scheme
• Technical challenges
• Evaluation
• Conclusions
Limitations of the “Straw man”

1. $O(L)$ communication cost
2. Collisions
3. Malicious clients
Limitations of the “Straw man”

1. $O(L)$ communication cost
2. Collisions
3. Malicious clients
Challenge 1: Bandwidth Efficiency

In “straw man” design, client sends DB-sized vector to each server

**Idea:** run PIR protocol *in reverse* to write into DB while sending fewer bits

PIR-in-reverse used in Ostrovsky-Shoup ’97 in single-client context

We extend their results to a many-client context (with malicious clients)
(k,t)-Distributed Point Functions

- We use a generalization of “DPFs” defined by Gilboa and Ishai (2014)
- Many one-round-trip PIR protocols construct DPFs implicitly

Goal: $|q_i| \ll |x_i|$ for all $i$

$(q_1, \ldots, q_k) \leftarrow \text{KeyGen}(m, \ell)$

$x_i \leftarrow \text{Eval}(q_i) \in \mathbb{F}^L$

$m \in \mathbb{F}; \ell \in [L]$
(k,t)-Distributed Point Functions

Correctness:

\[ (q_1, \ldots, q_k) \leftarrow \text{KeyGen}(1^n) \]

\[ m \cdot e_\ell \leftarrow \sum_{i=1}^{k} \text{Eval}(q_i) \]

(k,t)-Privacy: [In a minute]

Sum of the Eval() outputs will be zero everywhere, except at position \( \ell \).
DPF Correctness

\[(m, \ell)\]

\[\text{KeyGen} \rightarrow q_1 \rightarrow \text{Eval} \rightarrow x_1 \]

\[q_2 \rightarrow \text{Eval} \rightarrow x_2 \]

\[\ldots \rightarrow \ldots \rightarrow \ldots \]

\[q_k \rightarrow \text{Eval} \rightarrow x_k \]

\[= \]

\[
\begin{array}{cccccc}
0 & 0 & m & 0 & 0 & 0 \\
\end{array}
\]
(k,t)-Distributed Point Functions

(k,t)-Privacy: Can simulate the distribution of any subset $S$ of at most $t$ DPF keys

$$(q_1, \ldots, q_k) \leftarrow \text{KeyGen}(m, \ell)$$

$$\{q_i\}_{i \in S} \approx_c \text{Sim}(S)$$

$\forall S \subset [k]$

s.t. $|S| \leq t$

[Intuition: $t$ keys leak nothing about $m$ or $\ell$]
DPFs Reduce Bandwidth Cost
DPFs Reduce Bandwidth Cost
Alice sends $\sum_{i=1}^{k} |q_i|$ bits
Challenge 1: Bandwidth Efficiency

We show: a \((k, t)\)-private DPF

yields

a \(k\)-server write-anonymous DB scheme tolerating up to \(t\) malicious servers

I will present a \((2,1)\)-DPF with \(O(L^{1/2})\)-length keys based on PIR of Chor and Gilboa (’97)
(2,1)-DPF Construction

**Idea:** – Represent Eval() output as a matrix
  – Keys can be length of side
(2,1)-DPF Construction

KeyGen(·)
(2,1)-DPF Construction

**Idea:**
- Represent the evaluation output as a matrix.
- Keys can be based on the matrix index.

Output will sum to $m$ at $\ell = (i, j)$.
(2,1)-DPF Construction

Using

Key = (w, k_1, k_2, k_3, k_4, k_5), where each has length $O(\sqrt{L})$

Sampled at random
(2,1)-DPF Construction

G() is a PRG mapping keys \( k \) to \( L^{1/2} \) bits
(2,1)-DPF Construction
(2,1)-DPF Construction

Outputs are equal everywhere except at row 2

<table>
<thead>
<tr>
<th></th>
<th>k_1</th>
<th>k_2</th>
<th>k_3</th>
<th>k_4</th>
<th>k_5</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- \( G(k_1) \)
- \( G(k_2) + v \)
- \( G(k_3) + v \)
- \( G(k_4) \)
- \( G(k_5) + v \)

<table>
<thead>
<tr>
<th></th>
<th>k_1</th>
<th>k_2</th>
<th>k_3</th>
<th>k_4</th>
<th>k_5</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- \( G(k_1) \)
- \( G(k_2^*) \)
- \( G(k_3) + v \)
- \( G(k_4) \)
- \( G(k_5) + v \)

\( v \)

\( v \)
(2,1)-DPF Construction

Outputs sum to zero everywhere except at row 2

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>1</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$k_1$</td>
<td>$k_2$</td>
<td>$k_3$</td>
<td>$k_4$</td>
<td>$k_5$</td>
</tr>
<tr>
<td>$G(k_1)$</td>
<td>$G(k_3) + v$</td>
<td>$G(k_4)$</td>
<td>$G(k_5) + v$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$v$</td>
<td>$v$</td>
<td>$v$</td>
<td>$v$</td>
<td>$v$</td>
<td>$v$</td>
</tr>
</tbody>
</table>

$G(k_2^*)$
Construct $\mathbf{v}$ as:

$$\mathbf{v} = G(k_2) + G(k_2^*) + m \cdot e_j$$
\[ G(k_1) + G(k_2) + v \]
\[ G(k_3) + v \]
\[ G(k_4) \]
\[ G(k_5) + v \]

\[ \begin{align*}
G(k_1) \\
G(k_2^*) \\
G(k_3) + v \\
G(k_4) \\
G(k_5) + v
\end{align*} \]

\[ = \]

\[ \begin{align*}
00000\ldots00000 \\
0000000m000 \\
00000\ldots00000 \\
00000\ldots00000 \\
00000\ldots00000
\end{align*} \]
Challenge 1: Bandwidth Efficiency

- Brings comm cost down to $O(L^{1/2})$
  - Just requires PRG — fast!
- Recursive application of the same trick
  - Key size down to polylog($L$) [GI’14]
New DPF Construction

Given a seed-homomorphic PRG

\[ G(s_1) + G(s_2) = G(s_1 + s_2) \]

we build a \((k, k-1)\)-private DPF

\[ \text{[NPR'99]} \ [\text{BLMR'13}] \ [\text{BP'14}] \ [\text{BV'15}] \]

\[ \rightarrow \text{Privacy holds even if all but one server is adversarial} \]
Limitations of the “Straw man”

1. $O(L)$ communication cost
2. Collisions
3. Malicious clients
Challenge 2: Collisions

- Clients pick write location $\ell$ at random
- Two honest clients may write into the same location $\ell$
Challenge 2: Collisions

- Clients pick write location \( \ell \) at random
- Two honest clients may write into the same location \( \ell \)
Challenge 2: Collisions

- Clients pick write location $\ell$ at random
- Two honest clients may write into the same location $\ell$
Challenge 2: Collisions

• Clients pick write location $\ell$ at random
• Two honest clients may write into the same location $\ell$

Instead of getting $m_A$, $m_B$, get the sum

$m_A + m_B$

0
0
$m_A + m_B$
0
0
Challenge 2: Collisions

Straightforward solution:
Make DB table large enough to avoid collisions

Better solution:
Use coding techniques to recover from up to $d$-way collisions

Key idea: even after a collision, learn the sum of colliding writes

$$c = m_1 + m_2$$
Challenge 2: Collisions

**Idea**: To handle 2-collisions, can code message \( m \) as: \((m, m^2)\)

\[ \text{[Let } \text{char}(\mathbb{F}) > 2 \] \]

After a 2-collision, DBs recover the values:

\[
\begin{align*}
  c_1 &= m_1 + m_2 \\
  c_2 &= m_1^2 + m_2^2
\end{align*}
\]

**Given** \( c_1 \) and \( c_2 \) can recover \( m_1 \) and \( m_2 \)
Challenge 2: Collisions

Using coding technique, can tolerate $d$-collisions for any $d$.

For 1% loss rate, 1k users:

- Naive method: 100k cells
- Coding method: 6.9k cells

Reduces table size by 93%
Limitations of the “Straw man”

1. O(\(L\)) communication cost
2. Collisions
3. Malicious clients
Challenge 3: Malicious Clients

One malicious client can corrupt the entire DB!
Goal: Prevent evil client from destroying DB

- One way to solve this is with NIZKs
  - Expensive public-key crypto [Golle Juels ‘04]

- More efficient solution:
  - Add a third non-colluding “audit” server to get honest majority
  - Fast, info-theoretic MPC techniques [GMW’87], [CCD’88], [FNW’96]
DB Server X

0 1 1 0 1

k₁ k₂ k₃ k₄ k₅

v

DB Server Y

0 0 1 0 1

k₁ k₂ k₃ k₄ k₅

v
DB Server X

0 | k₁
1 | k₂
1 | k₃
0 | k₄
1 | k₅

DB Server Y

0 | k₁
0 | k₂*
1 | k₃
0 | k₄
1 | k₅
DB Server X

DB Server Y

Auditor

\[ a_1 \quad a_2 \quad a_3 \]

\[ b_1 \quad b_2 \quad b_3 \]
Auditor

DB Server X

DB Server Y

offset

\[ a_1 \quad a_2 \quad a_3 \]

\[ b_1 \quad b_2 \quad b_3 \]
Auditor

\[ h_2(a_2), h_3(a_3), h_1(a_1) \]

\[ h_2(b_2), h_3(b_3), h_1(b_1) \]

DB Server X

\[ h_1, h_2, h_3 \]

DB Server Y
$h_2(a_2)$  $h_3(a_3)$  $h_1(a_1)$

$DB Server X$

$h_2(b_2)$  $h_3(b_3)$  $h_1(b_1)$

$DB Server Y$
Equal almost everywhere?
Auditor

DB Server X

DB Server Y
Outline

• Motivation
• Definitions and a “Straw man” scheme
• Technical challenges
• Evaluation
• Conclusions
Implementation

• Implemented the full protocol in Go
  – 2 DB servers + 1 audit server

• Ran perf evaluation on a network testbed simulating real-world net conditions
Bottom-Line Result

• For a DB with 65,000 Tweet-length rows, can process **30 writes/second**
• Can process **1,000,000 writes** in 8 hours on a single server

⇒ Main bottleneck is PRG expansion
At large table sizes, PRG cost dominates.
Outline

• Motivation
• Definitions and a “Straw man” scheme
• Technical challenges
• Evaluation
• Conclusions
Open Problems

1. Reduce $\Theta(L)$ computation cost at server
   – Using multiple rounds per write?

2. Key-homomorphic DPFs
   – Another way to reduce cost at server

3. $(k, k-1)$-private DPFs without PKC
   – Possible without seed-hom PRGs?
Conclusion

In many contexts, “hiding the metadata” is as important as hiding the data.

Combination of crypto tools with systems design → 1,000,000-user anonymity sets.

“Multi-user writable PIRs” have applications to private messaging.
  – Still ∃ barriers to practicality (+ open problems)
Questions?