Publius: A robust, tamper-evident, censorship-resistant web publishing system
By Waldman, Rubin, and Cranor

Presented by Marco Barreno
October 8th, 2003
CS 294-4: Peer-to-peer systems

Speech, censorship, and anonymity

- Speech, publication spread ideas
  - Many ideas are controversial, some revolutionary
- People in power like to stay in power
- Those in power may try to censor ideas or persecute people they see as threats

...and the Internet...

- “The Net treats censorship as damage and routes around it” --John Gilmore
- “On the Internet, nobody knows you're a dog,” caption to a New Yorker cartoon by Peter Steiner
- Do these sentiments accurately represent the state of the internet?

...and Reality.

- Church of Scientology
  - Among other incidents, tried to force Julf Helsingius to reveal identities anonymized through remailer
- DMCA
  - Copyright owners can force service providers to remove content (must stay down during dispute)
- Generally, an automatic, semipermanent record of much internet activity is (or could be) kept
### Publius design goals

- **Core goals:**
  - Censorship resistant
  - Tamper evident
  - Deniable
  - Fault-tolerant
  - Persistent
  - Freely available
- **Additional goals:**
  - Source anonymous; updateable; extensible

### Related tools

- Anonymizers
- Onion routing
- Crowds
- Rewebber
- Freenet
- Eternity Service

### Eternity Service (Ross Anderson)

- **General problem:** assure availability
  - Most work focused on confidentiality and integrity
- **Eternity service:** persistent, available data
  - Can’t be deleted by anyone, including servers and author
- **Threat model:** governments ban it, attack it
  - But law tends to lag technology
  - Need strong cryptography, wide distribution

### Freenet

- **Very similar in goals:** defeat censorship, provide anonymity
- **Different approaches:** routing vs. lookup, anonymous servers vs. key shares, ...
- **We’ll come back to this later**
Overview of Publius

- Publishers, servers, retrievers
- Supports static content
- Assume static systemwide list of servers
- Each server doesn’t know what it hosts
- Key split into shares
- Special Publius URL
- Authors (and only authors) can update, delete

K-way key splitting

- Goal: split key K into n shares such that any k of them can recover the key but any k-1 cannot
- The strategy: find a polynomial q of degree k-1
- Evaluate q at n points (q(1), q(2), ..., q(n))
- q(0) = K, the other q(i) are the shares
- Can interpolate polynomial to find q(0) with any k points, but k-1 points give no information!

Example: $k = 2 \ (q(i) = m*i + q(0))$

Publishing

- Alice encodes content M with key K: $\{M\}_K$
- She splits K into n shares (k can recover K)
- For each share: $name_i = wrap(H(M, share))$
  $location_i = (name_i \ mod \ m) + 1$
- Why wrap (xor of halves)? To save space?
- If fewer than d unique locations are obtained, start over with another K
Publishing (continued)

- What is d?
  - d is the number of Publius servers that will end up hosting the content (k <= d <= m)
  - Coupon collector’s problem: expect y ln(y) coupons
  - Therefore we choose d and then set n = d ln(d)
- At each location i: publish \( \{M\}_K \), share \( \lambda_i \), and some other information in directory \( \text{name}_i \)
- Publius URL comprises at least d concatenated \( \text{name}_i \)s

Retrieving

- Bob wants to view content; he gets Publius URL
- He gets the \( \text{name}_i \)s and computes location \( \text{location}_i \)
- Choose k servers arbitrarily: get \( \{M\}_K \) from one and get shares from all k
- Recover K, decrypt, and check \( \text{name}_i \)s
- If there’s a problem, try different k shares

Retrieving (continued)

- More on problems...
  - If the name \( i \) calculation works, either the data are uncorrupted or a collision has been found in SHA-1!
  - If something does go wrong, Bob can try a different set of k shares, up to \( (n \choose k) \) combinations
  - Other options: Berlekamp-Welch, brute-force search of \( n^*(n \choose k) \) share/document combinations
  - A note: Bob can be retrieving the content through a normal web browser with a Publius proxy

Deleting

- Alice, the author, creates passwords \( H(\text{servername}, PW) \) from a master PW
  - Specific to each server
- Alice authenticates with password to delete
- Problem...?
Deleting

- Alice, the author, creates passwords \( H(\text{servername}, \text{PW}) \) from a master \( \text{PW} \)
  - Specific to each server
- Alice authenticates with password to delete
- Problem...?
  - The password is sent in the clear!
  - Should public keys be passed with server list?

Update

- Optional 'update' file in name\( _i \) directory
  - Redirects to updated version
  - If 'update' exists, retrievals follow it
- Uses same password as delete
- But this indirection seems very kludgy

Update

- Optional 'update' file in name\( _i \) directory
  - Redirects to updated version
  - If 'update' exists, retrievals follow it
- Uses same password as delete
- But this indirection seems very kludgy

Mutually linked documents

- Problem: Publius names are based on content, so a dependency loop can't be resolved directly
- Solution: Use an update
## Limitations and threats

- **Share deletion/corruption**
  - Cannot recover key if \( n-k+1 \) shares are toast
  - Higher \( n \) and lower \( k \) lead to more protection against censorship

- **Updates**
  - Malicious servers can add/change/delete 'update' files
  - Need enough servers collaborating to do real damage

## More threats

- **Denial of service**
  - A malicious user could saturate Publius with junk
  - Solution: Hash Cash?

- **No connection-based anonymity**

- **'Rubber-Hose cryptanalysis’**
  - Unlike Eternity, authors can delete: open to coercion
  - Coercing enough servers difficult if well distributed

## Other potential problems

- Could a server be held legally negligent for not knowing what it hosts?
  - It could find out by obtaining the Publius URL and retrieving the document, or querying enough other servers

- Lack of connection-based anonymity is especially bad here
  - Any one of the \( n \) servers can compromise an author’s anonymity

## Back to Freenet

- **Key similarities:**
  - Both aim to keep publishers anonymous
  - Both aim to preserve content

- **Key differences:**
  - Freenet: adversary doesn’t know where content is;
    Publius: server doesn’t know what it stores

  - What about resistance to censorship?
That’s all folks!