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

Ranjit Randhawa
Department of Computer Science
Virginia Tech
Outline

- Overview of Publius
- Anonymity tools
- Publius in detail
  - Publish
  - Retrieve
  - Delete
  - Update
  - Example
- Publishing issues
- Limitations
Overview of Publius

♦ Web publishing system
♦ Highly resistant to censorship
♦ Provides publishers with a high degree of anonymity
♦ Social issues
  – Censorship
  – DMCA
Overview cont’d

♦ Publius consists of:
  – Publishers: who post Publius content to the web
  – Servers: who host random-looking content
  – Retrievers: who browse Publius content on the web
Anonymity tools I

♦ Connection based anonymity tools
  – Provide connection based anonymity

♦ Tools
  – Anonymizer (Proxy server)
  – Mix models
  – Onion Routing
  – Crowds
  – Freedom Anonymity System
Anonymity tools II

♦ Author based Anonymity tools
  – Hide location or author of a web document

♦ Tools
  – Janus/Rewebber
    • Combination connection/author based tool
    • Anonymizer + URL rewriting service
  – Andersons Eternity Service
    • Server based storage
    • Once copied, never removed
  – FreeNet
    • Adaptive network, stores files locally in nodes (which db)
  – Intermemory
    • Focus is on preservation of electronic media
Publius - Publish

- Encrypt content $M$ with key $K$ (produces $\{M_i^k\}$) and split $K$ into $n$ shares using Shamir secret sharing
  - Any $k$ of them can reproduce the secret
- Compute 8 byte long name for all $n$ (for addressing)
  - $name_i = \text{wrap}(H(M.share_i))$ [$H = \text{cryptographic hash}$]
- Computer location ($m = \text{servers on system}$)
  - $location_i = (name_i, \text{MOD}m) + 1$
  - Obtain $n$ values between 1 and $m$ iff $d >= k$ ($d = \text{min number of servers that will hold the Publius content}$)
- Publish Publius $\{M\}k.share_i$ in dir $name_i$, on servers at locations $location_i$ (index into list of servers)
- URL contains $d name_i$ values concatenated together
Publius - Retrieve

- From Publius URL, $U$, parse out $name_i$ values
- $location_i = (name_i, MODm) + 1$
- Index in table of servers for each of the shares
- Choose $k$ arbitrarily and retrieve encrypted file
  - File named $file$ stored on each server in $name_i$ directory
- Retrieve other $k-1$ shares
- Combine all shares to form key $K$ and decrypt file
- Verify all $name_i$ values are correct (regenerate $U$)
  - If not, try different set of $k$ shares
Publius - Delete

♦ Before publishing password $PW$ generated
♦ Send encrypted document, share and hash of domain name concatenated with $PW$ to servers that will host the document
♦ To delete send hash of domain name concatenated with $PW$ to each server along with $name_i$ that corresponds to that server
♦ Server removes directory matching the $name_i$ and all its files.
Publius - Update

♦ Enable publisher to change content without changing the URL
♦ Publisher specifies filename (containing new content), original URL, original $PW$ and new $PW$.
♦ Update program first publishes new content
♦ Original URL used to fine $n$ servers that host original content.
♦ Each server then places the new URL in the update file and deletes the contents of the old file
♦ When user requests file that has been updated the servers return the update URL instead of the contents
Example Publius URL

♦ Publius URL
  – http://!anon!/options encode(name$_1$)…encode(name$_n$)

♦ Encode function generates ASCII representation of \textit{name\textsubscript{i}} value

♦ Options
  – 2 characters that define how Publius client software interpret URL (as there are no file extensions)
  – 16 bit options section encodes
    • Version number, number of shares needed and update flag
Publishing Issues

♦ Publish files A and B
  – Where A contains link to B
  – Where A and B contain links to each other
♦ Publish a directory
Limitations

- Share deletion/corruption
- Update file deletion/corruption
- DoS attacks
- Threats to publisher anonymity
- Rubber-Hose cryptanalysis