Combating Web Plagiarism and Improving Internet Safety by Authenticating Web Content

Vivek Pathak
vpathak@cs.stevens.edu
Plagiarism on the Web

- **World Wide Web**
  - Important source of information
  - Publishing content is easy
  - Easy to plagiarize
    - Readers can not determine the original copy

- **Source of income for bloggers and others**
  - Plagiarism dis-incentivizes content creation
    - Quality
    - Correctness
Web Content Authentication

- **Authenticate web content**
  - Identify the *original copy* on the web
  - Automatic cryptographic solution
    - Able to handle automated plagiarism
    - Detect and correct web plagiarism

- **Computer security problem of practical importance**
  - Content is migrating online
  - Increasing importance
    - Economically
    - In terms of credibility
Web Content Issues

- Web browsing
  - Important source of malware infection
  - Hacked web sites
  - DNS attacks

- Authenticate original authorship of content
  - Avoid phishing by forwarding users to correct destination
Improving Web Safety

- Additional use of the content authentication mechanism
  - Certify content as malicious or benign
    - Based on commodity web page classification tools

- Robust and timely compared to URL based approach
  - For example, Google safe browsing API
  - Improve web safety
Technical Challenges

- Identifying plagiarism
  - Scale of the Web
    - More than 10B pages and growing
    - Plagiarism detection must be timely
      - Automated plagiarism may be virtually instantaneous

- Correcting plagiarism
  - Must be automatic
    - Legal procedures are slow compared to speed of the web
  - How to prove authorship
Web Authentication Plugin

- Insert authentication functionality into commodity browsers
  - Collaborative protocol establishes original authorship
  - Implement peer-to-peer storage
    - Store deductions about web pages
      - Original copy
      - Harmful web page

- Use the content authentication to allow safe browsing of the original web
Web Authentication Plugin

- Browser with authentication plugin
  - Detect and avoid plagiarized and malware web pages
  - Co-operate to form a peer to peer storage
    - Store information about web content
Web Authentication Outline

- New web page
  - Decide if original, plagiarized, or malware
  - Store status in peer-to-peer storage
    - Digital signature with the self generated public key authenticated using the Byzantine fault tolerant public key authentication protocol (BPKA)

- User browses existing web page
  - Query the peer-to-peer storage for page information
    - Chord style peer-to-peer storage (DHT)
  - Allow if the current content is already authenticated
  - Else incrementally re-authenticate new content
Web Authentication Idea

- Visiting a web page
  - Each visitor signs and dates content
    - Content signature and creation date
    - Store digitally signed record on highly available peer-to-peer storage

- Handle changing web content
  - Content signatures degrade gracefully
    - Incremental benign changes are tolerated
    - Malicious content changes are not
- Web pages accessed through URLs
  - Use normalized forms of URLs

- Web browsers
  - Identify through IP addresses
  - Operate DHT and BPKA protocols
  - Save multiple authentication certificates for a give URL to content association
Public Key Authentication

- Use Byzantine fault tolerant public key authentication protocol
  - Authenticate public key with an automatic peer-to-peer protocol
  - Based on honest majority assumption
  - Decentralized and eventually correct
  - Details in
    - **SGKA**: Social-group key authentication for email. *Fifth Conference on Email and Anti-Spam, August 2008.*
Detecting Similar Web Content

- Web page content
  - Separate into text and non-text information
  - No approximate matching for non-text

- Allow approximate text match using Broder’s shingles algorithm
  - Calculate shingles, i.e. substrings in the document
  - Generate a content fingerprint by min-wise selection of shingles’ hashes

- Content fingerprint matches on high resemblance
Detecting Similar Web Content

from Broder 2000
Storage Design

- Distributed hash tables
  - Store $V[k]$ at chord node successor to $\text{Hash}(k)$

- Chord maintains
  - Finger tables for efficient routing
  - Replication for robustness
  - Storage location determined by secure hash function
  - Minority of malicious nodes can be tolerated

(from Wikipedia)
Data Structures

- DHT stores key to value pairs
  - Key value based data structure for content authentication

<table>
<thead>
<tr>
<th>URL INFORMATION</th>
<th>Url</th>
<th>Original/Plagiarized/Malware Status</th>
<th>[Plagiarized from Url]</th>
<th>Timestamp</th>
<th>Certificates List</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>CONTENT FINGERPRINT</th>
<th>Url</th>
<th>Content digest</th>
<th>List of sentence hashes</th>
<th>List of shingles</th>
<th>List of additional content digests (formatting, images, etc.)</th>
<th>Timestamp</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>CONTENT OWNERSHIP</th>
<th>Hash value</th>
<th>List of &lt;URL, Timestamp&gt; ordered by oldest timestamps first</th>
</tr>
</thead>
</table>
Authentication Algorithm

- **Common case**
  - Operate through the URLInformation DHT

- **If not known to be malicious**
  - Check content fingerprint
  - Open if the page is identical

---

Algorithm 1: Web content authentication algorithm

```
Input: URL U being visited by the user
Result: Detect if the web content associated with U is malware, plagiarized, or original good content.

1. if UrlInformation[U] is Malicious then
2.     return null
3. end
4. if UrlInformation[U] is Plagiarized then
5.     return UrlInformation.PlagiarizedFrom[U]
6. end
7. Content ← content(U)
8. Fingerprint ← fingerprint(Content)
9. if UrlInformation[U] is Original then
10.    if Fingerprint = ContentFingerprint[U] then
11.        return U
12.    end
13. end
```
Authentication Algorithm

- Upon finding new page or content
  - Duplicate detection used for identify plagiarism

- Publishing
  - Run insertion before making content visible on the web

```c
// Found new URL or modified content
if isMalware(FingerPrint) is true then
    UrlInformation[U] ← Malicious;
    return null;
end

Candidates←null;
foreach Content hash h ∈ FingerPrint do
    if ContentOwnership[h] is not null then
        Candidates.append(ContentOwnership[h]);
    end
end

V ← GetPlagiarismSource(Candidates);
if V is null then
    ContentOwnership.Update(Fingerprint);
    ContentFingerprint[U].Update(Fingerprint);
    return U;
else
    UrlInformation[U] ← Plagiarized;
    UrlInformation.PlagiarizedFrom[U] ← V;
    return V;
end
```
Engineering Considerations

- Using peer-to-peer storage
  - Allow scalable authentication at the web scale

- Chord lookups take log(n) steps
  - At most 30 steps to query from storage operating on 300M browsers
  - Can speed up more using up some extra memory

- Each node responsible for k/n keys
  - For 10B web size
  - Store information for 10B/300M or 33 pages
Advantage of Web Authentication

- Users can determine which copy is original

- Incentive for publication
  - Original authors get advertising revenues for content

- Can build tools to improve web safety
  - Hacked web sites
  - DNS poisoning attacks
  - Different content than what was authenticated by peers
Conclusion

- Content authentication
  - Detect plagiarism automatically
  - Reasonable overhead per peer

- Shared and signed content authentication
  - Protect against hacked web sites
  - Handle DNS poisoning attack
Detecting Hacked Web Sites

How web sites are designed
- Content management system
- Web site infrastructure
  - PHP, Ruby, etc.

Attack model
- Exploit vulnerability in infrastructure or tools
- Add malicious code to website
- Infect visitor browser or computer

Hacked web site is not the original published
Defense against DNS Poisoning

- What are DNS attacks
  - Domain resolves to incorrect IP
  - Useful for malicious activity
    - Phishing for passwords
    - Receiving ad revenue

- Recent examples
System Model

- Mutually authenticating peers
  - Associate network end-point with public key
  - Asynchronous network
    - No partitioning
    - Eventual delivery after retransmissions
- Disjoint message transmission paths
  - Man-in-the-middle attack on Ø fraction of peers
Attack Model

- Malicious peers
  - Honest majority
  - At most $t$ of the $n$ peers are malicious, where $t = (1 - 6\varnothing)/3 \cdot n$
  - Threshold of malicious peers ($t$) follows from the analysis shown in the following slides

- Adversaries
  - Relax network-is-the-adversary model
  - Limited power to prevent message delivery
    - Indicated by the constant $\varnothing$
  - Unlimited power to spoof
Authentication Model

- Challenge-response protocol
  - Authenticate public key
  - Use encrypted nonce as challenge
  - No active attacks

- Man in the middle attack
  - Limited number of attacks $O(\emptyset)$

- Proof of possession of public key $K_A$
  $$P_{BA} = \{B,A,\text{Challenge},K_A(r)\}_B, \{A,B,\text{Response},r\}_A$$
Authentication Model (contd.)

- Distributed authentication
  - Challenge response from multiple peers
  - Gather proofs of possession

- Lack of consensus on authenticity
  - Malicious peers
  - Man-in-the-middle attack
Authentication Correctness

- **Validity of proofs of possession**
  - $P_{EA} = \{E,A,\text{Challenge},K_A(r)\}_E, \{A,E,\text{Response},r\}_A$
  - Nonce is recovered correctly
  - Digital signatures on messages

- **Recent proofs stored by peers**
  - Signed messages
  - Prove malicious behavior

<table>
<thead>
<tr>
<th>Proofs at B</th>
<th>$P_{BA}$</th>
<th>$P_{CA}$</th>
<th>$P_{DA}$</th>
<th>$P_{EA}$</th>
<th>$P_{FA}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Proofs at A</td>
<td>$P_{BA}$</td>
<td>$P_{CA}$</td>
<td>$P_{DA}$</td>
<td>$P_{EA}$</td>
<td>$P_{FA}$</td>
</tr>
</tbody>
</table>
Byzantine Agreement

- Publicize lack of consensus
  - Share proofs of possession

- Each peer tries to authenticate A
  - Sends its proof-of-possession vector to every peer
  - Byzantine agreement on authenticity

- Majority decision at every peer
  - Identify malicious peers
  - Complete authentication

<table>
<thead>
<tr>
<th>From</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>From</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>From</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>From</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>From</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>From</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Byzantine Agreement
Correctness

- Consider proofs received at a peer

Set of peers participating in the authentication protocol

$\Phi_n$ on compromised path to $A$

$\Phi_n$ on compromised path to peer

t malicious peers
Byzantine Agreement

Correctness (contd.)

- Byzantine Agreement
  - $t + 2\delta n$ proofs may not arrive
    - Peer receives at least $n-t-2\delta n$ proofs
  - $t + 2\delta n$ proofs may be faulty
    - Peer receives at least $n-2t-4\delta n$ correct agreeing proofs
    - Peer decides correctly by majority if $n-2t-4\delta n > t + 2\delta n$
  - Agreement is correct if $t < (1-6\delta)/3 n$

- Theorems proved in research paper
  - Honest peers are authenticated correctly if there is an honest majority
  - Honest majority is preserved
Authentication with Trusted Groups

- Reduce messaging cost

- Trusted group
  - Authenticate public keys

- Probationary group

- Un-trusted group
  - Known to be malicious

- Details in research paper
  - Probability of incorrect authentication drops exponentially with time