Towards Append-Only Authenticated Dictionaries

Vivek Bhupatiraju, CS-PRIMES 2017
Public-key Cryptography

\[ M \xrightarrow{e(M, PK)} \text{ciphertext} \xrightarrow{SK} M \]
Secure Channels

- Having secure channels is becoming more and more necessary
- Many of these systems based around public-key cryptography
- Essential to accurately distribute and access these public-keys
- Let’s use a directory!
John publishes his public key, $PK_j$

Directory stores $PK_j$ under John's name

Robert sends $MS_R$

Robert encrypts $MS_R$ with $PK_j$

John decrypts with $SK_j = \Rightarrow MS_R$

$e(MS_R, PK_j)$
John publishes his public key, $PK_J$.

Directory stores $PK_M$ under John's name, sends $PK_J$ to Mark.

Directory sends Robert $PK_M$.

Robert encrypts $MS_R$ with $PK_M$.

Mark now knows $MS_R$ - no secrecy.

John decrypts with $SK_J \Rightarrow MS_R$.

Mark encrypts $MS_R$ with $PK_J$.

Robert now knows $MS_R$. 

John encrypts $MS_R$ with $PK_M$.

Mark now knows $MS_R$ - no secrecy.
Detecting Impersonation! (NON-MEMBERSHIP)

Directory

John = PK_J
Alin = PK_A
•
•
•

1 Needs to check that directory is not hiding a PK_M under his name

2 Sends cryptographic proof that this IS the case

John
PK_J + SK_J
Detecting Impersonation! (CONSISTENCY)

Directory

John = PK_J
Alin = PK_A

1. Needs to check that directory is not hiding a PK_M under his name

2. Sends cryptographic proof that this IS the case
Detecting Impersonation! (MEMBERSHIP)

1. Sends Robert $PK_M$ when he asks for John’s PK

2. Asks for proof that $PK_M$ is in the directory under John’s name

Robert $MS_R$

Directory

John = $PK_J$
Alin = $PK_A$

•

•
Append-Only Dictionaries (Key-value pairs)

- **NON-MEMBERSHIP**
  - Proof that no values exist for key $n$ other than the ones in the tree

- **CONSISTENCY**
  - Proof that all data in version $i$ of the dictionary is also in version $j$ of the dictionary, where $i \leq j$

- **MEMBERSHIP**
  - Proof that $(n, v_n)$ is in dictionary
## Attempts at a Full AAD

$m = \text{number of key-value pairs in AAD / Server}$

<table>
<thead>
<tr>
<th></th>
<th>Membership</th>
<th>Non-membership</th>
<th>Consistency</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>History Tree</strong></td>
<td>$O(\log(m))$</td>
<td>$O(m)$</td>
<td>$O(\log(m))$</td>
</tr>
<tr>
<td><strong>Prefix Tree</strong></td>
<td>$O(\log(m))$</td>
<td>$O(\log(m))$</td>
<td>$O(m)$</td>
</tr>
<tr>
<td><strong>Quadratic Prefix Forest</strong></td>
<td>$O(\sqrt[3]{m} \times \log(m))$</td>
<td>$O(\sqrt[3]{m} \times \log(m))$</td>
<td>$O(\log(m))$</td>
</tr>
</tbody>
</table>
Merkle Tree

\[ H(A) \]
\[ A = (a, v_a) \]

\[ H(B) \]
\[ B = (b, v_b) \]

\[ H(C) \]
\[ C = (c, v_c) \]

\[ H(D) \]
\[ D = (d, v_d) \]

\[ H(H(A) \mid H(B)) = L \]

\[ H(H(C) \mid H(D)) = R \]

\[ H(L + R) = \Omega \]
(Merkle Root)
History Tree

- Just a merkle tree that grows as key-value pairs are added to it
History Tree (MEMBERSHIP)

Merkle Root: $\Omega_o =? \Omega_c$

Space/Time Complexity $O(\log(m))$
History Tree (NON-MEMBERSHIP)

Space/Time Complexity

\[ O(m) \]
History Tree (CONSISTENCY)

Space/Time Complexity: $O(\log(m))$
Prefix Tree

Tree defined by hashes:

- HASH('a') = 1100...
- HASH('b') = 0011...
- HASH('c') = 1010...
- HASH('d') = 0001...

Also a merkle tree!
- Each node is a hash of its children
Prefix Tree \((\text{MEMBERSHIP})\)

Space/Time Complexity \(O(\log(m))\)
**Prefix Tree** (NON-MEMBERSHIP)

HASH(‘e’) = 0011...

Space/Time Complexity \(O(\log(m))\)
Prefix Tree (CONSISTENCY)

- Server has to send all key-value pairs added between versions OR membership proofs
  - Both linear in complexity, O(m)
Quadratic Prefix Forest

$U_1$ of Forest, Size 1

$U_2$ of Forest, Size 4

$U_n$ of Forest, Size $n^2$

tree of size $n^2$
Quadratic Prefix Forest

- Say there are \( n \) trees in the forest

\[
\sum_{i=0}^{n} i^2 = \frac{(n)(n+1)(2n+1)}{6}
\]

- If there are \( m \) total key-value pairs

\[
\text{# of trees} = O\left(\sqrt[3]{m}\right)
\]
Q. Prefix Forests (MEMBERSHIP)

# of Trees: $O(\sqrt[3]{m})$

Space / Time Complexity: $O(\sqrt[3]{m} \times \log(m))$
Q. Prefix Forests (NON-MEMBERSHIP)

# of Trees: $O(\sqrt[3]{m})$

Space / Time Complexity: $O(\sqrt[3]{m} \times \log(m))$
Q. Prefix Forests (CONSISTENCY)

- Keep each of the Merkle roots of each prefix tree in a larger history tree
- Merkle roots of each prefix tree should never change
- Can check (via membership proofs) the roots of the prefix tree against those stored in the history tree
- Space/time complexity of $O(\log(m))$
Q. **Prefix Forests** *(USABILITY)*

- Each tree’s size is a square number
- At $m = 1,000,000$
  - Next tree will need ~10,000 new key-value pairs
- Sacrificing usability for better complexities in other operations
Future Work

- Algebraic Hashing
  - $H(a, b) = L \cdot a + R \cdot b$
- Bilinear Accumulators
  - Accumulating sets into small digests
  - Incorporating NON-MEMBERSHIP into history trees
- Coding up trees to test viability
- Exploring new data structures
Acknowledgements

I would like to thank:

- **Alin Tomescu**, my mentor
- **Srini Devadas**, coordinator of CS-PRIMES
- My parents and family
- MIT-PRIMES program
Thank you!

Ask me questions!