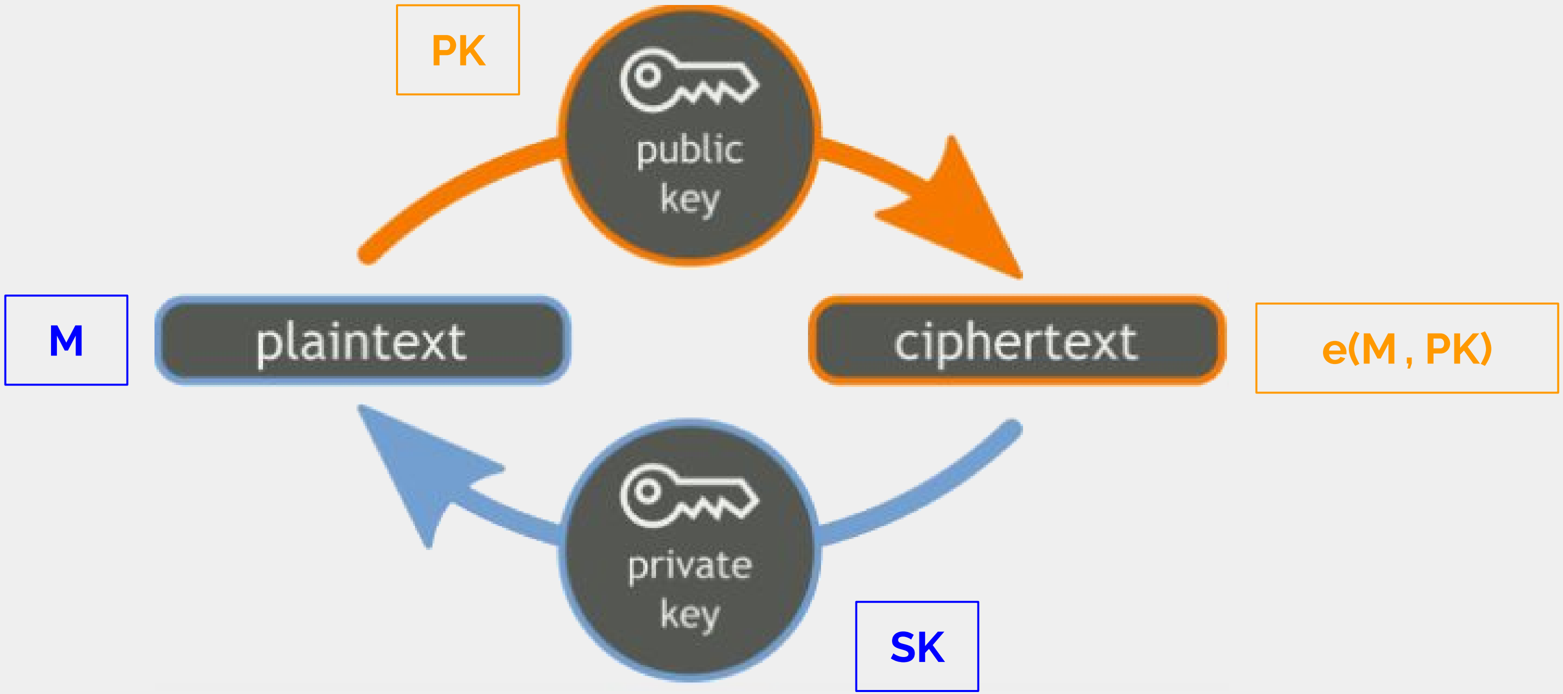
Towards Append-Only Authenticated Dictionaries

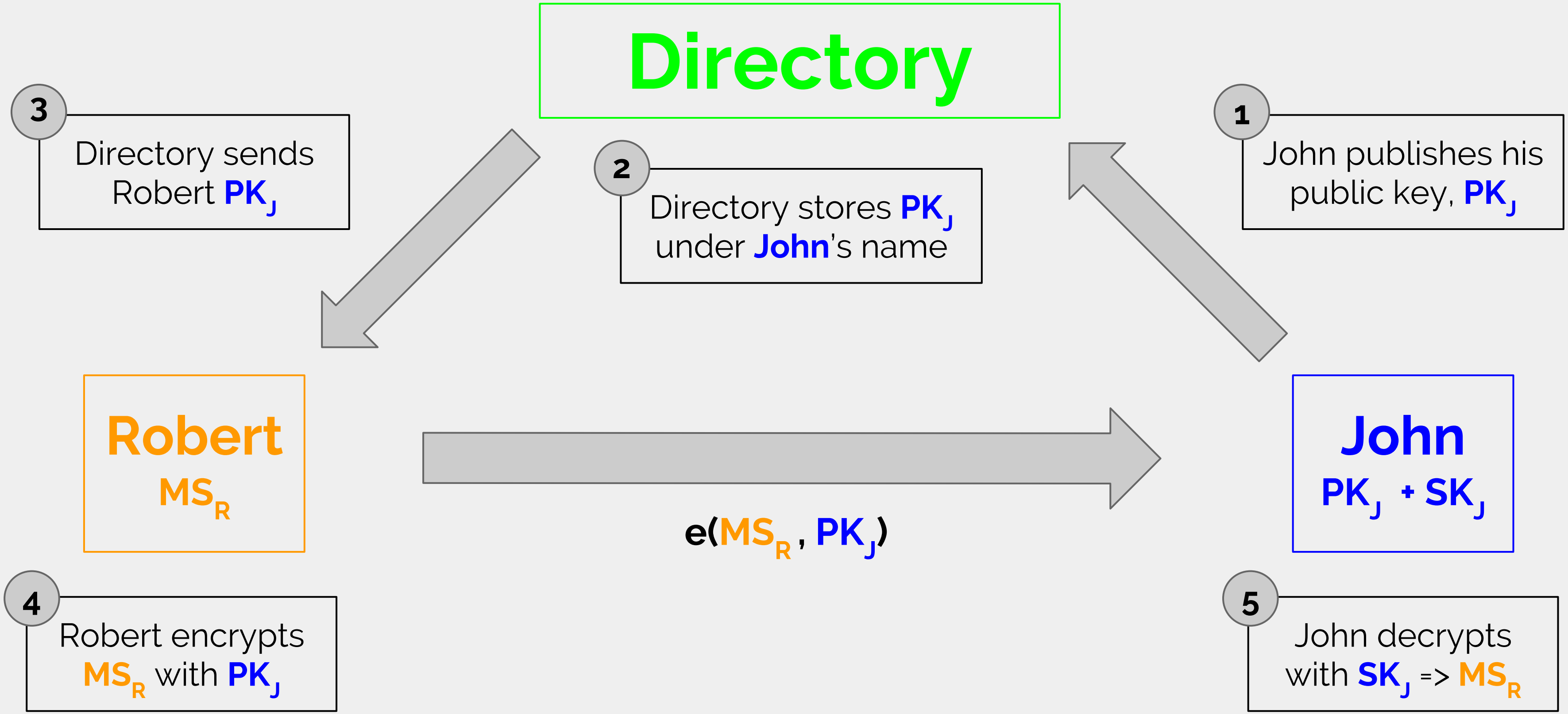
Vivek Bhupatiraju, CS-PRIMES 2017

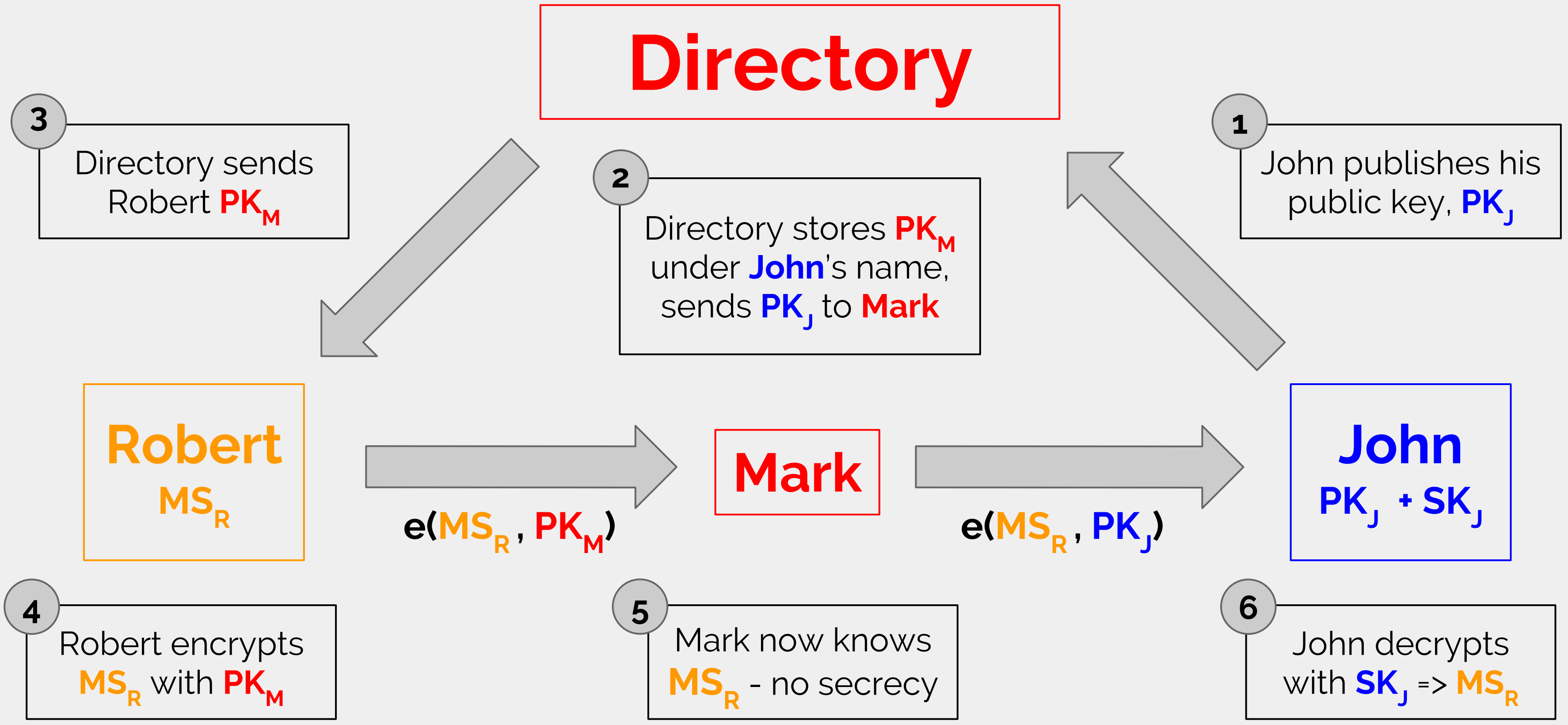
Public-key Cryptography



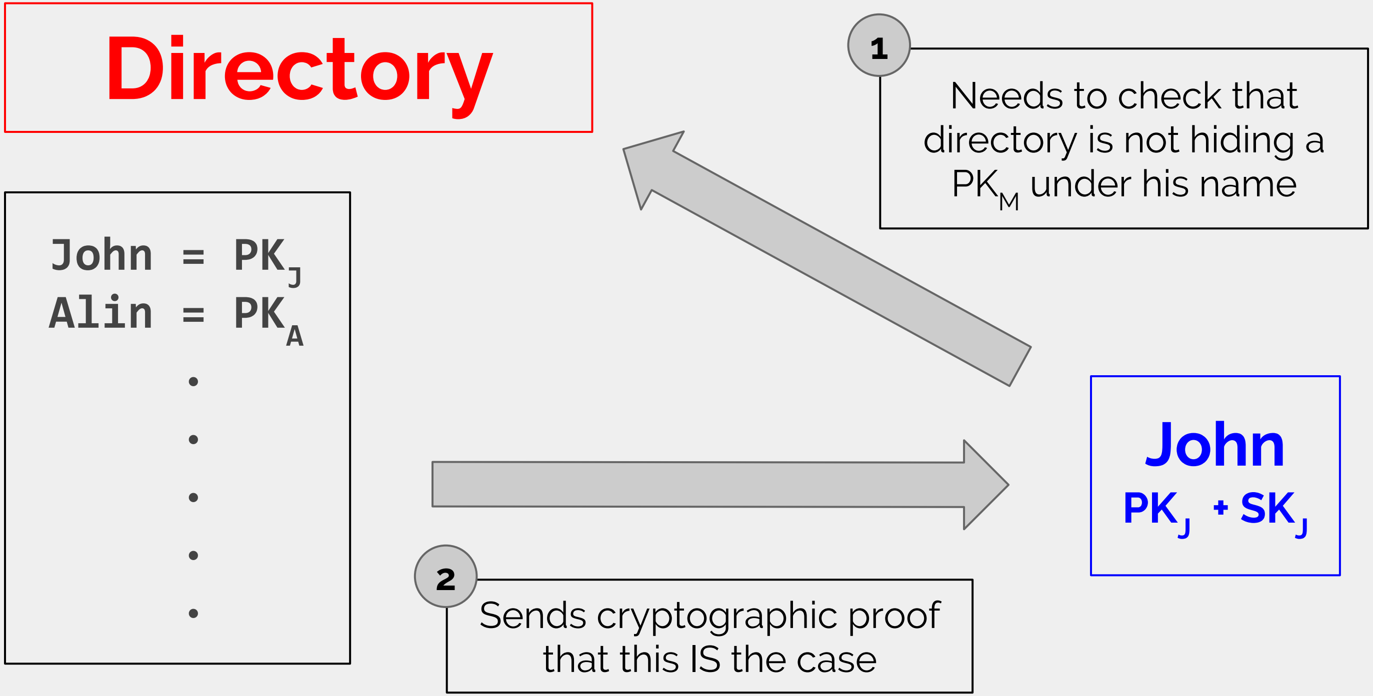
Secure Channels

- 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!


Directory
3
Directory sends Robert PK_J
2
Directory stores PK_J under John's name
1
John publishes his public key, PK_J
Robert
MS_R
$e(MS_R, PK_J)$
John
$PK_J + SK_J$
4
Robert encrypts MS_R with PK_J
5
John decrypts with SK_J => MS_R


Directory
3
Directory sends Robert PK_M
2
Directory stores PK_M under John's name, sends PK_J to Mark
1
John publishes his public key, PK_J
Robert
MS_R
e(MS_R , PK_M)
Mark
e(MS_R , PK_J)
John
PK_J + SK_J
4
Robert encrypts MS_R with PK_M
5
Mark now knows MS_R - no secrecy
6
John decrypts with SK_J => MS_R

Detecting Impersonation! (NON-MEMBERSHIP)



Detecting Impersonation! (CONSISTENCY)

Directory

John = PK_M
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! (MEMBERSHIP)

Directory

1

Sends Robert PK_M when he asks for John's PK

Robert

MS_R

2

Asks for proof that PK_M is in the directory under John's name

```
John = PK_J
Alin = PK_A
.
.
.
.
.
```

<u>Append-Only Dictionaries</u> (Key-value pairs)

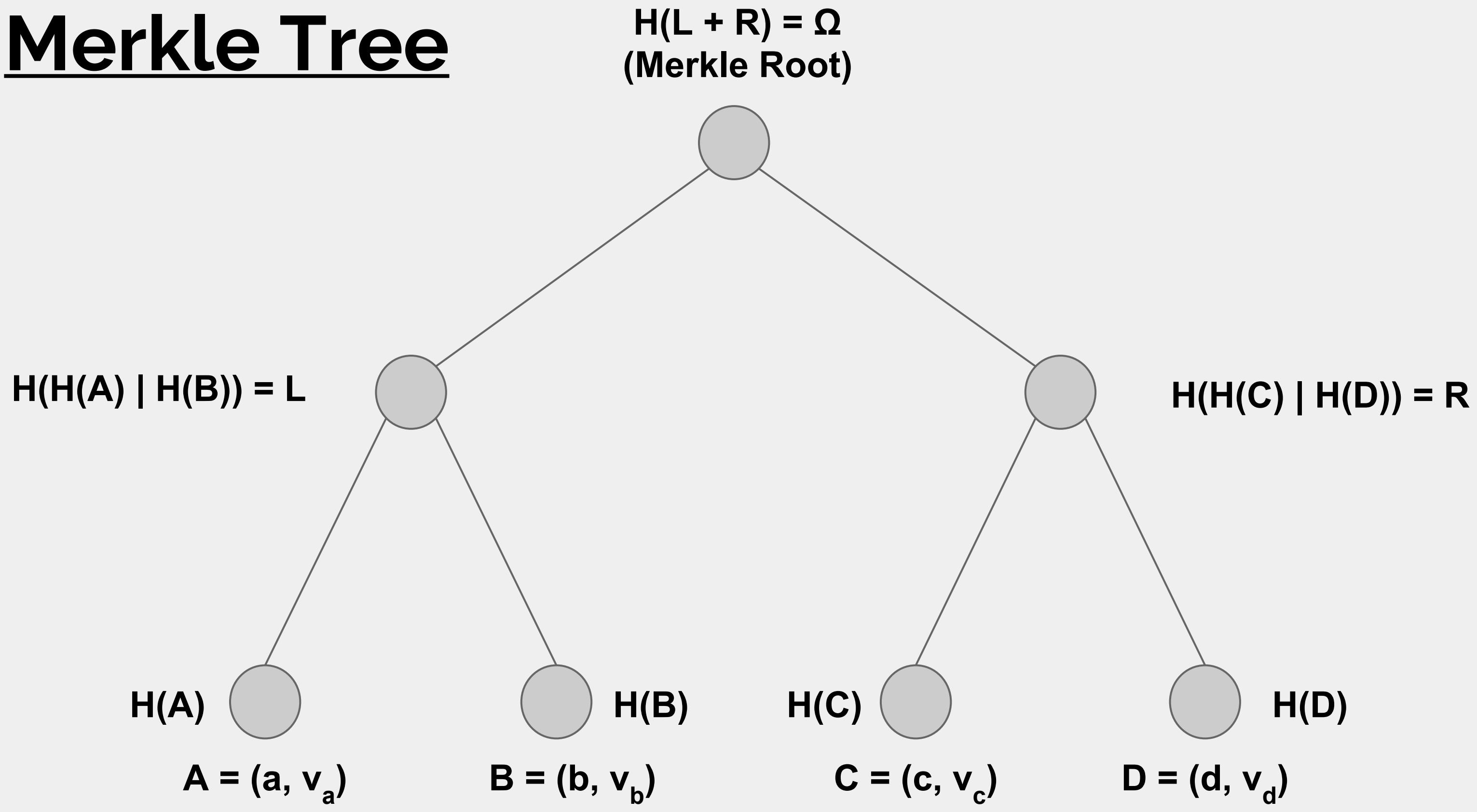
- **NON-MEMBERSHIP**
  - Proof that no values exist for key n other than the ones in the tree
- **CONSISTENCY**
  - Proof that all data in version i of the dictionary is also in version j of the dictionary, where $i \leq j$
- **MEMBERSHIP**
  - Proof that (n, v_n) is in dictionary

Attempts at a Full AAD

m == number of key-value pairs in AAD / Server

	Membership	Non-membership	Consistency
History Tree	$O(log(m))$	$O(m)$	$O(log(m))$
Prefix Tree	$O(log(m))$	$O(log(m))$	$O(m)$
Quadratic Prefix Forest	$O(\sqrt[3]{m} \times log(m))$	$O(\sqrt[3]{m} \times log(m))$	$O(log(m))$

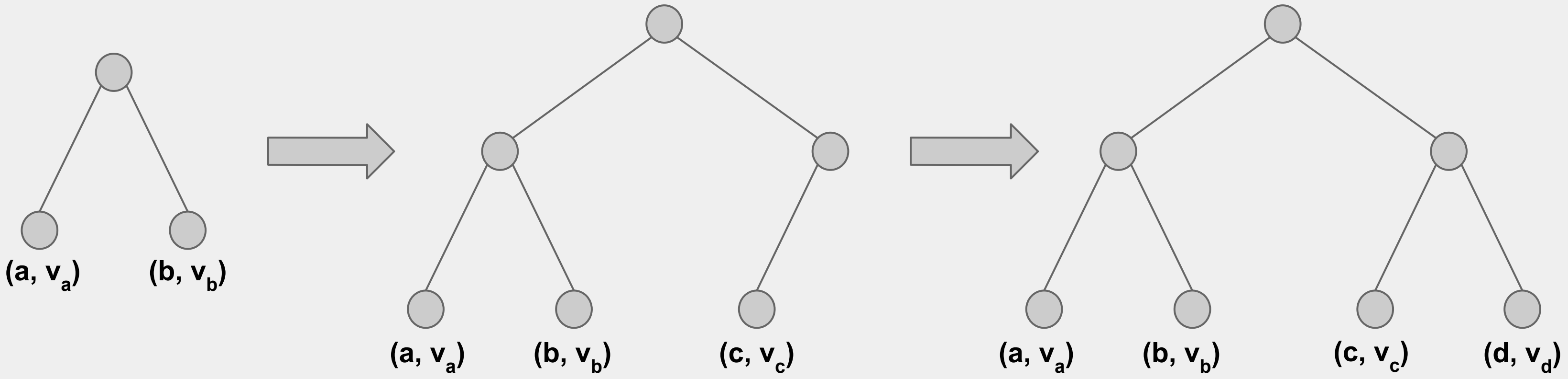
Merkle Tree



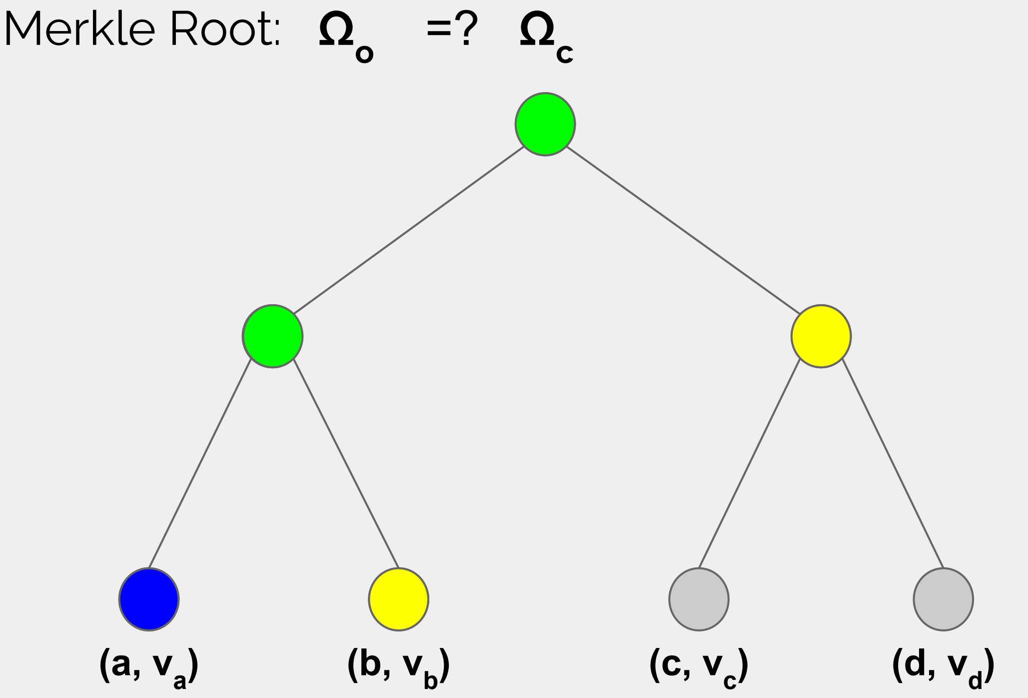
H(L + R) = Ω
(Merkle Root)
H(H(A) | H(B)) = L
H(H(C) | H(D)) = R
H(A)
H(B)
H(C)
H(D)
A = (a, v_a)
B = (b, v_b)
C = (c, v_c)
D = (d, v_d)

History Tree

- Just a merkle tree that grows as key-value pairs are added to it



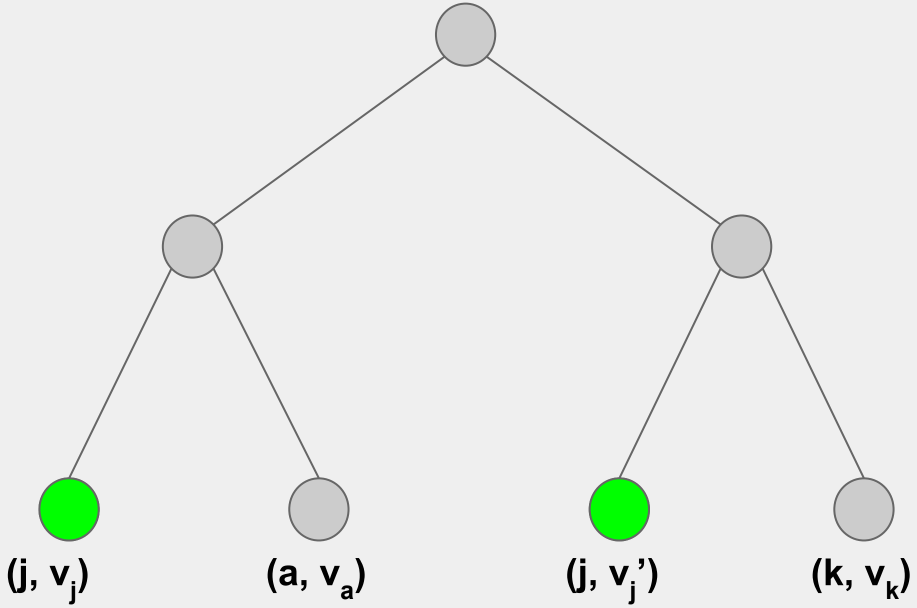
History Tree (MEMBERSHIP)



Space/Time Complexity

$O(log(m))$

History Tree (NON-MEMBERSHIP)

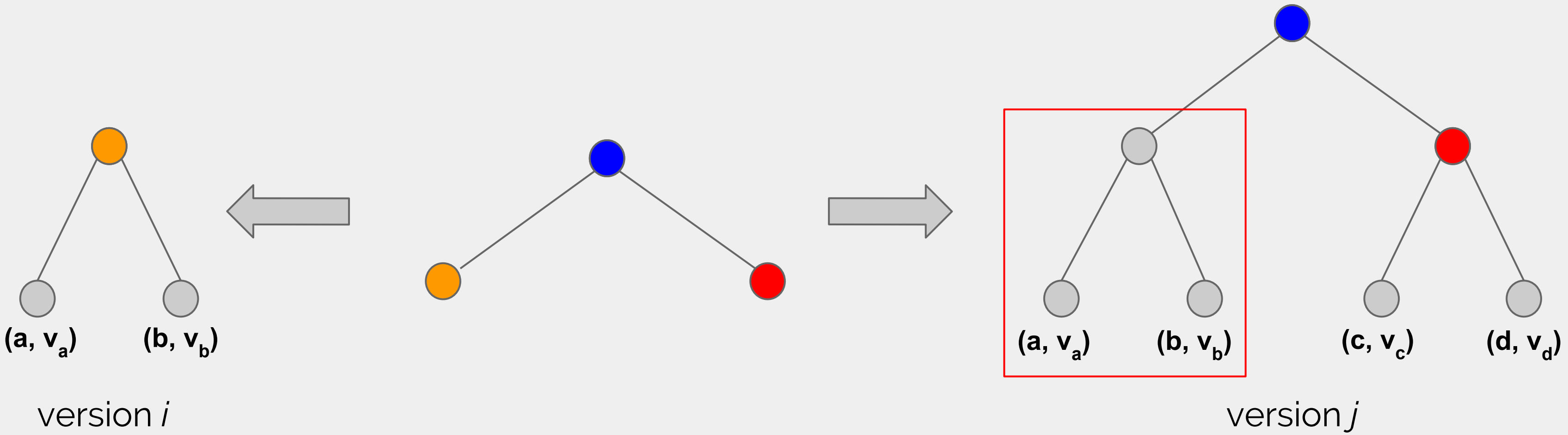


Space/Time Complexity

$O(m)$

History Tree (CONSISTENCY)

Space/Time Complexity: $O(log(m))$



Prefix Tree

Tree defined by hashes:

HASH('a') = 1100...

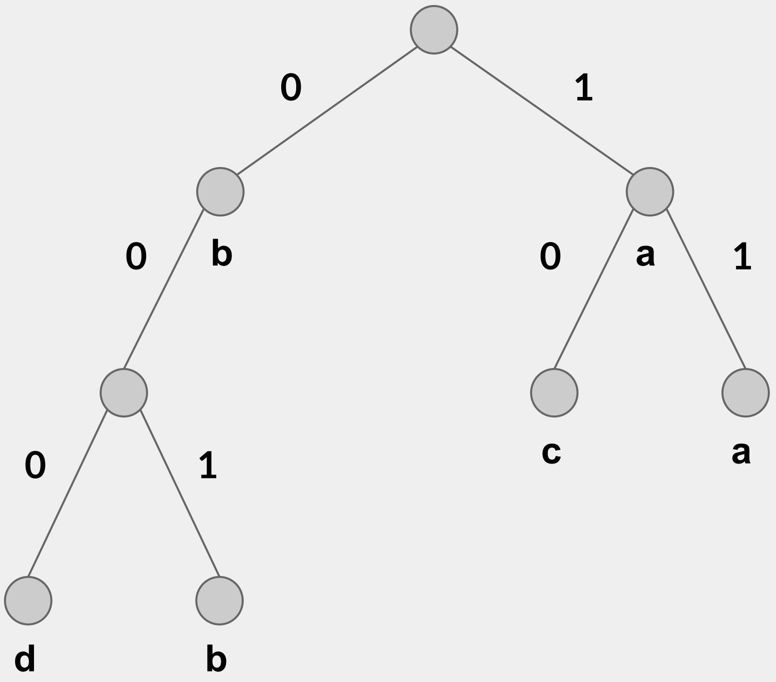
HASH('b') = 0011...

HASH('c') = 1010...

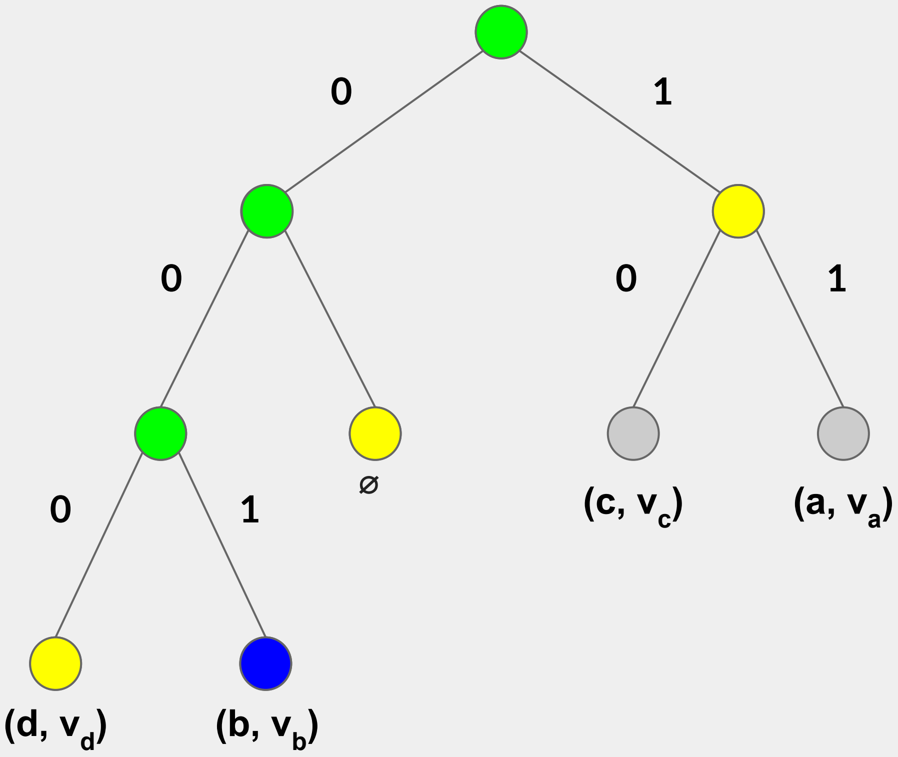
HASH('d') = 0001...

Also a merkle tree!

- Each node is a hash of its children



Prefix Tree (MEMBERSHIP)

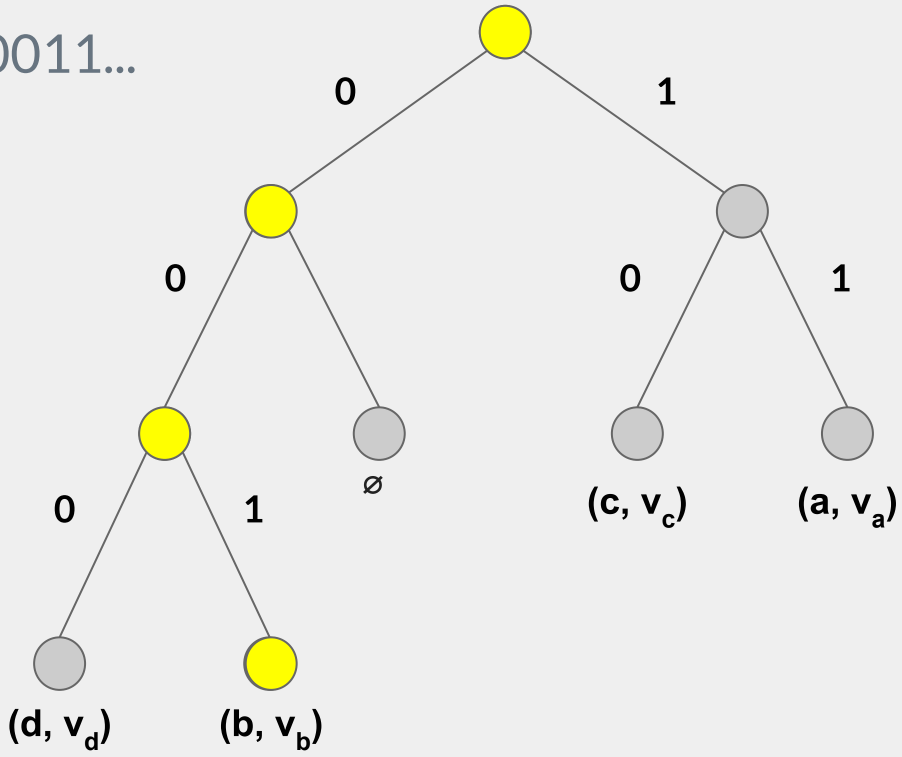


Space/Time Complexity

$O(log(m))$

Prefix Tree (NON-MEMBERSHIP)

HASH('e') = 0011...



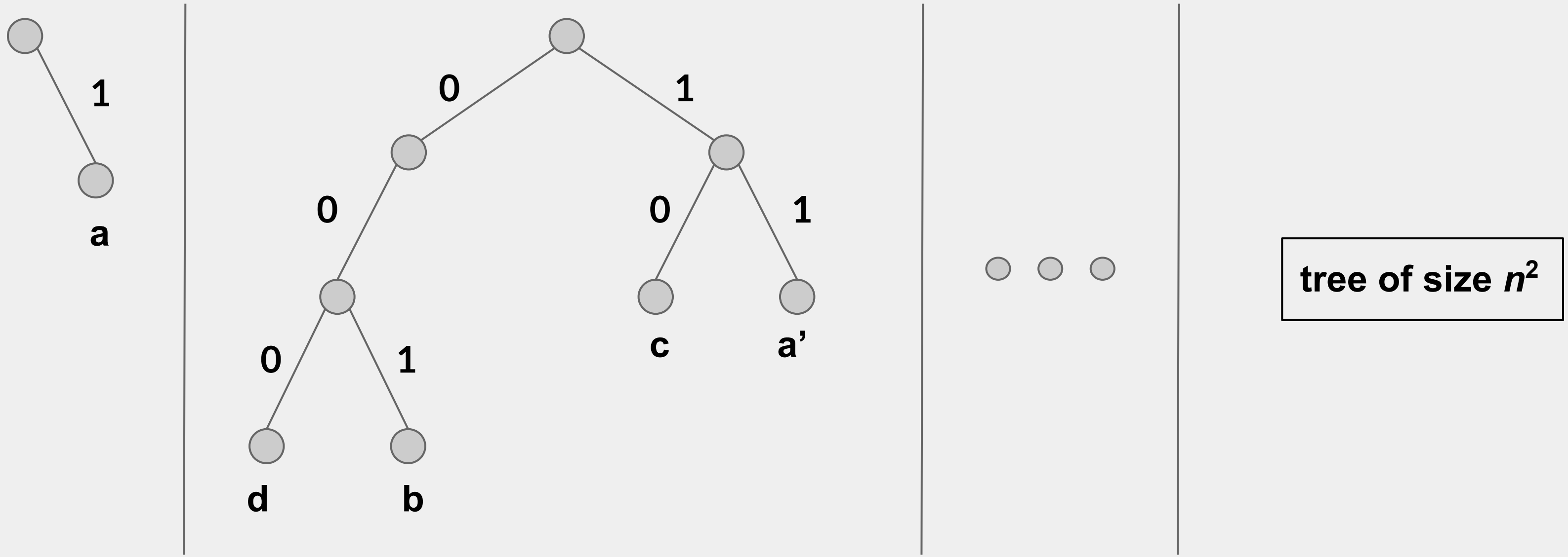
Space/Time Complexity

$O(log(m))$

<u>Prefix Tree</u> `(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

Quadratic Prefix Forest

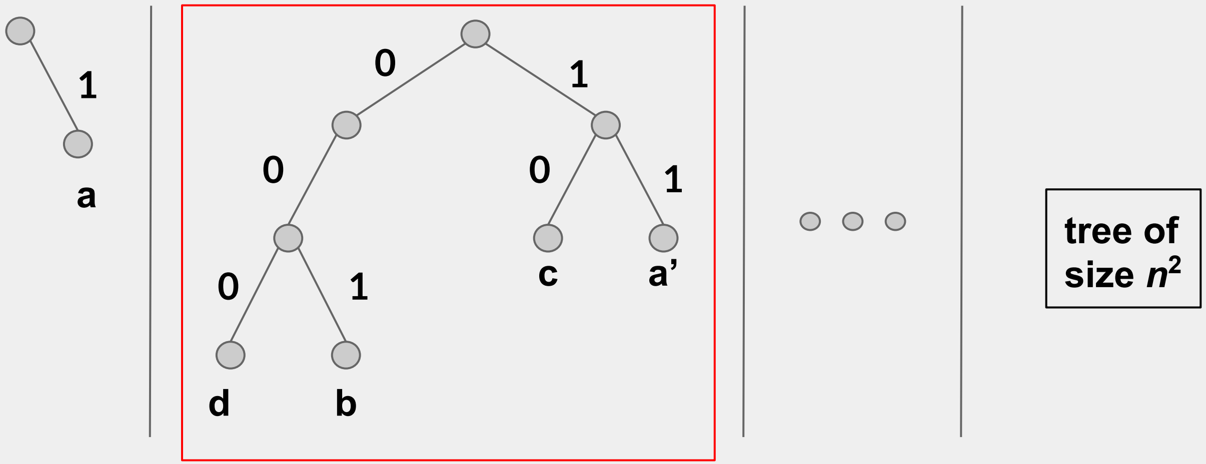
- Say there are n trees in the forest

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

- If there are m total key-value pairs

$$\#\ of\ trees = O(\sqrt[3]{m})$$

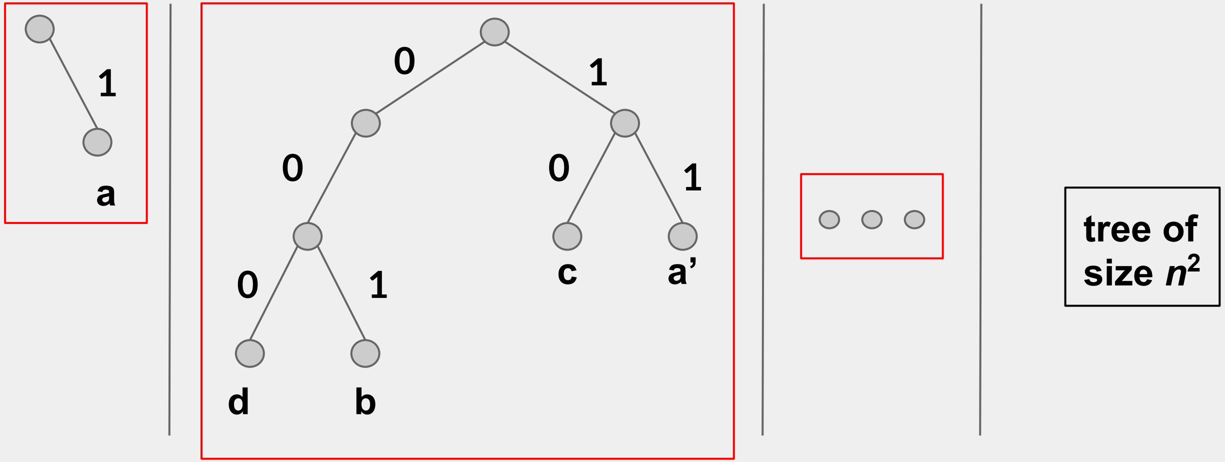
Q. Prefix Forests (MEMBERSHIP)



# 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 $O(log(m))$

<u>**Q. Prefix Forests**</u> `(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 * a + R * 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!