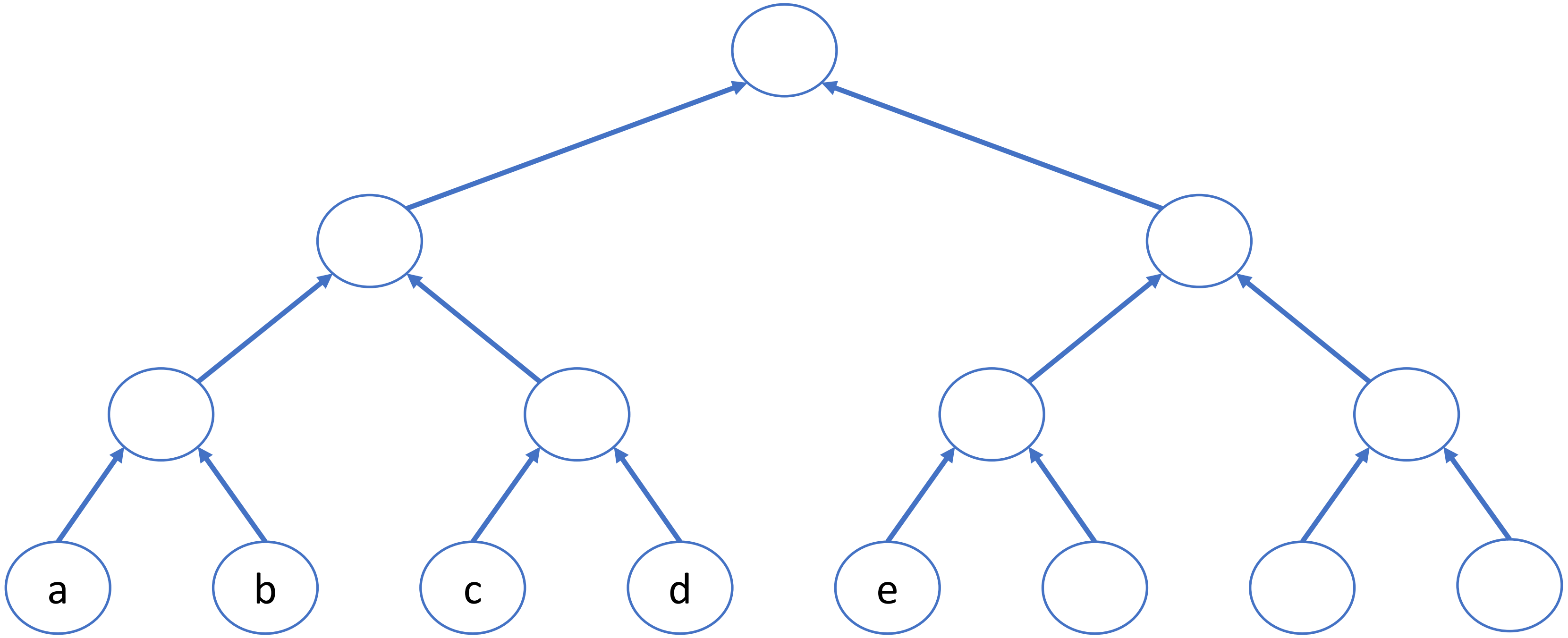
TreeKEM:
finding a balance between mKEM and ART

Bhargavan, Barnes, Rescorla, Beurdouche, Kobeissi, Naldurg, …

Tree-based Group Messaging



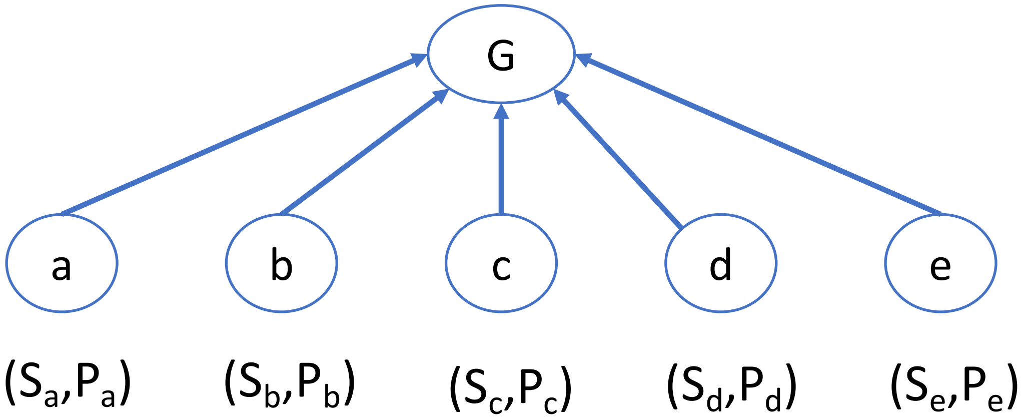
Goal: Efficiently and securely send a message m to {a,b,c,d,e}

mKEM: the naïve solution

N.P. Smart [2005]

Members = {a,b,c,d,e}

Key = $\{P_a,P_b,P_c,P_d,P_e\}$



SETUP

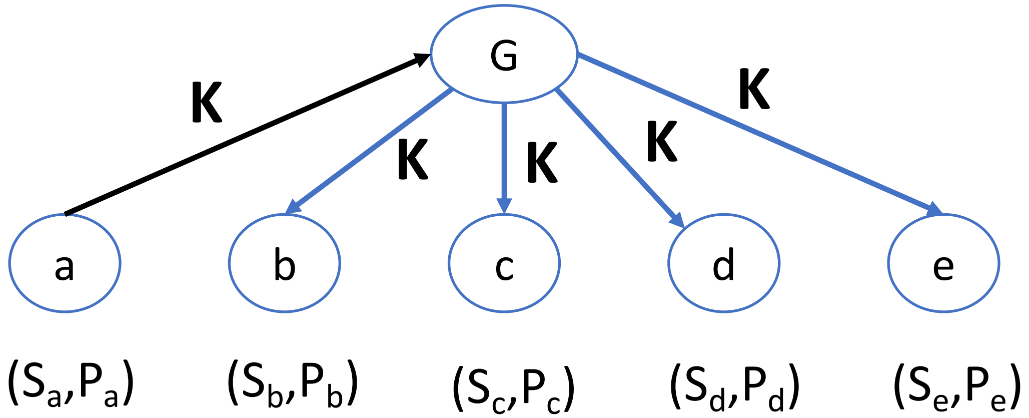
- Each participant has an encryption-decryption keypair
- Encryption keys $\{P_a,P_b,P_c,P_d,P_e\}$ are published to the group

MESSAGING

- *SEND(m):*
  encrypt m to each public key (n ENC)
- *RECV(m):*
  decrypt m using my public key (1 DEC)

mKEM: Setting up a Group Key

Members = {a,b,c,d,e}
Key = **K,** $\{P_a, P_b, P_c, P_d, P_e\}$



SETUP

- *SEND-CREATE:* (n ENC)
  a sends K encrypted to each public key
- *RECV-CREATE:* (1 DEC)
  others decrypt K with their public key

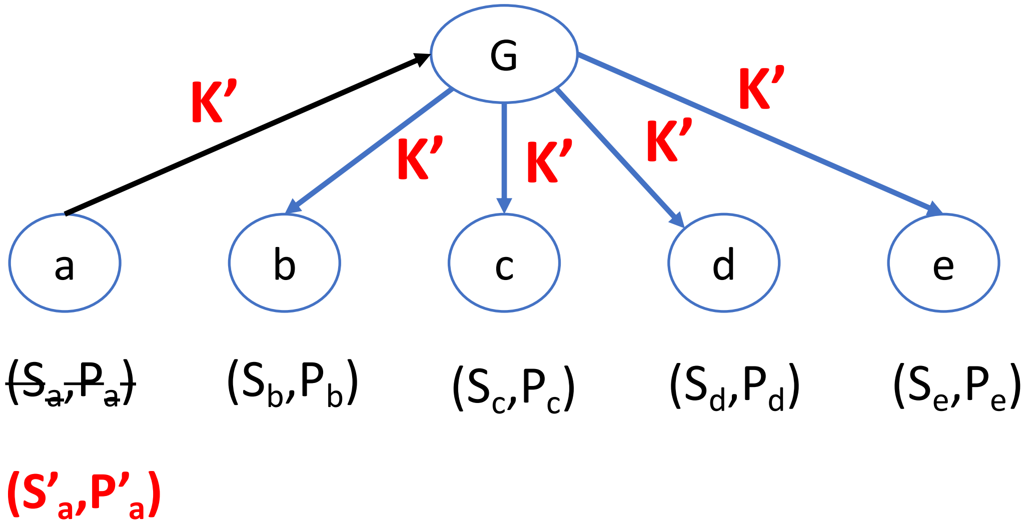
MESSAGING

- *SEND(m):* (1 ENC)
  encrypt m using K
- *RECV(m):* (1 DEC)
  decrypt m using K

Repeated mKEM: Supporting Dynamic Groups

Members = {a,b,c,d,e}

Key = **H(K,K')**, $\{P'_a, P_b, P_c, P_d, P_e\}$



UPDATE/REMOVE

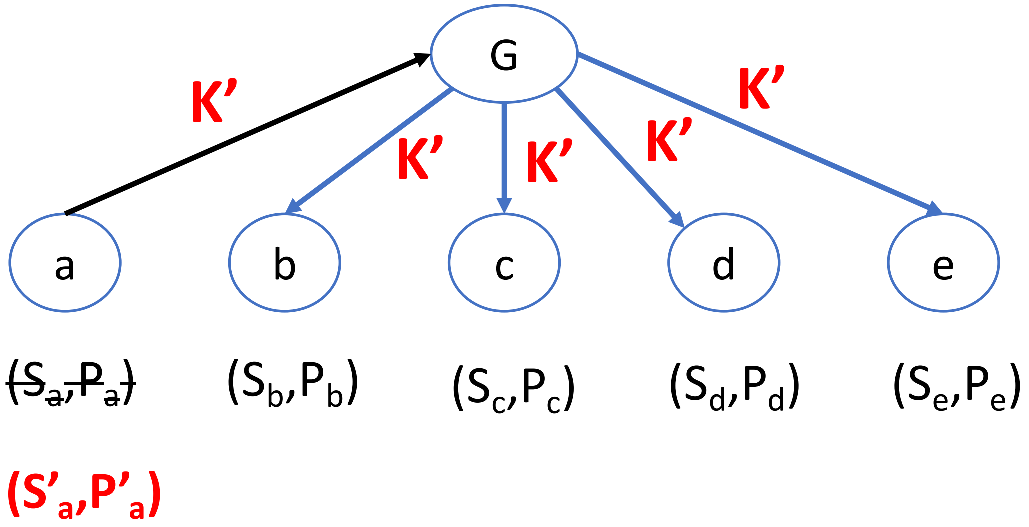
- *SEND-UPDATE:* (n ENC)
  a sends K encrypted to others
- *RECV-CREATE:* (1 DEC)
  others decrypt K with their public key

ADD

- *SEND-ADD(f):* (1 ENC)
  encrypt m using K, and P_f
- *RECV-ADD(f):* (1 DEC)
  decrypt m using K, or S_f

Repeated mKEM: Security Guarantees

Members = {a,b,c,d,e}
Key = **H(K,K')**, {P'_a,P_b,P_c,P_d,P_e}



SECRECY INVARIANT

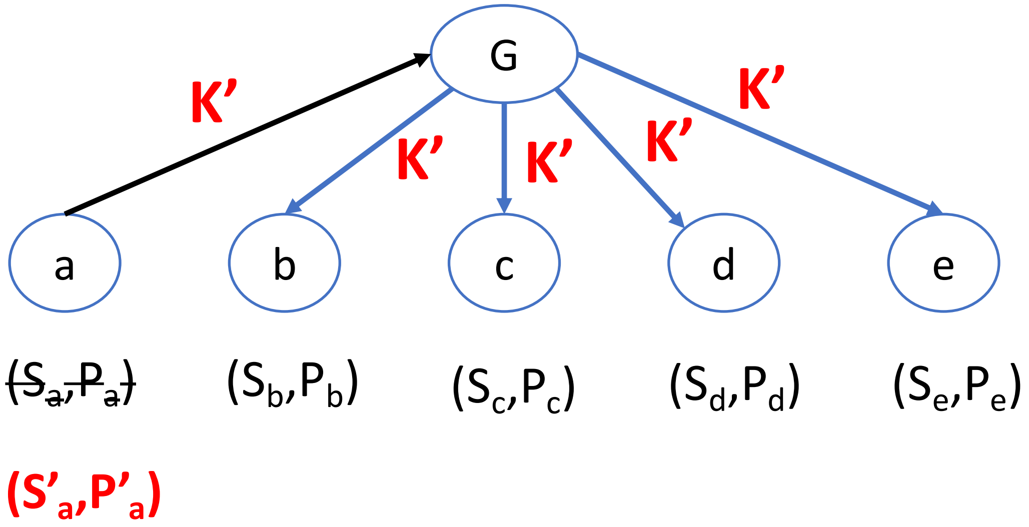
- *Only current owners of leaf decryption keys know the group key*
- (Authentication guarantees from Auth layer are orthogonal)

FORWARD/POST-COMPROMISE SECRECY

- *If all members regularly update their leaf keys and delete old keys,*
  we get PCS and FS.

Repeated mKEM: Additional Features

Members = {a,b,c,d,e}
Key = **H(K,K’),** $\{P'_a,P_b,P_c,P_d,P_e\}$



No Double-Join

- The creator/remover does not know any leaf keys except their own

Batched Changes

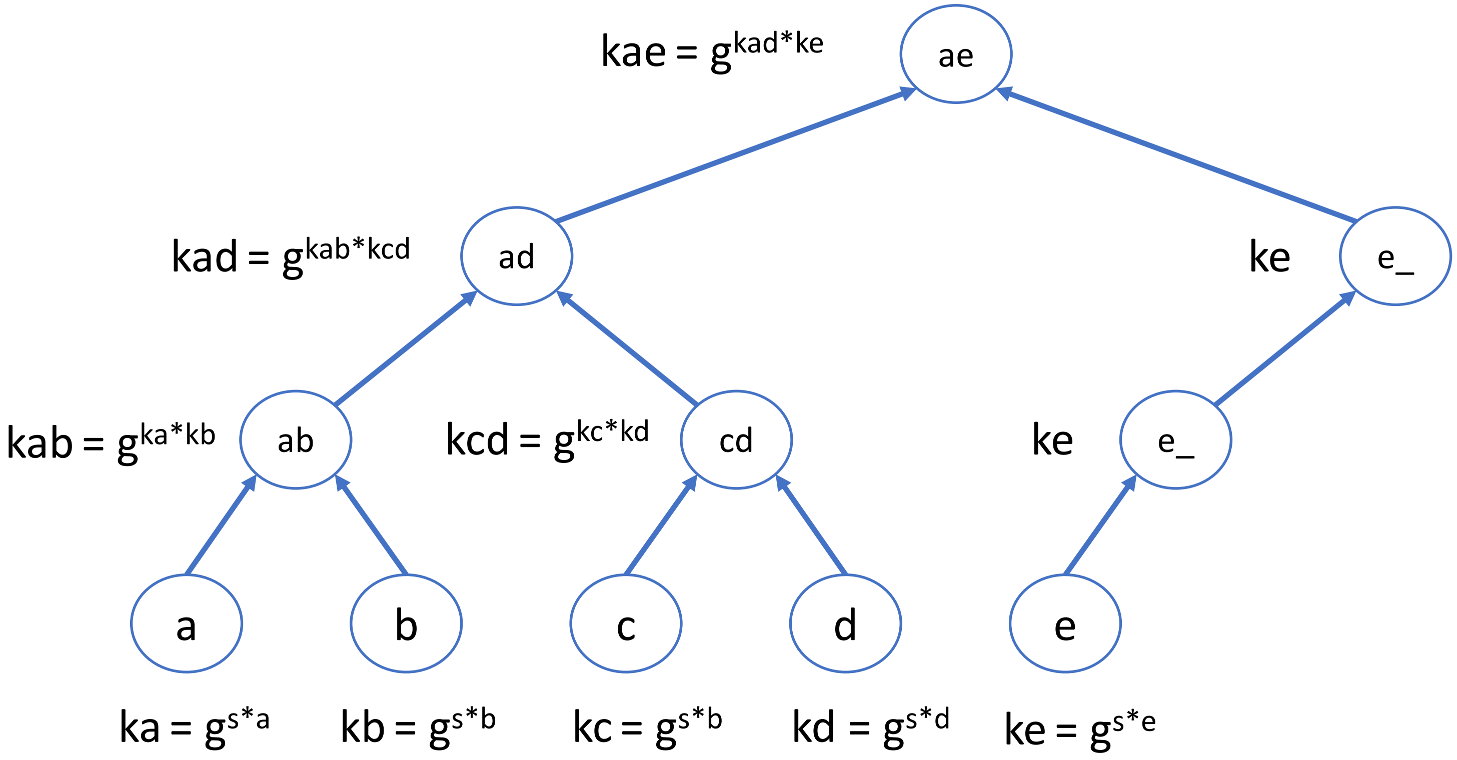
- Can batch k changes (ADD/REM/UPD) in a single update (n ENC, 1 DEC)

Merging Concurrent Changes

- Concurrent group key changes can be merged into a sequence.

Asynchronous Ratcheting Trees

Cohn-Gordon et al. [2018]



Before Setup

- *Send Create:* n DH ops
- *Recv Create:* log(n) DH ops

After Setup

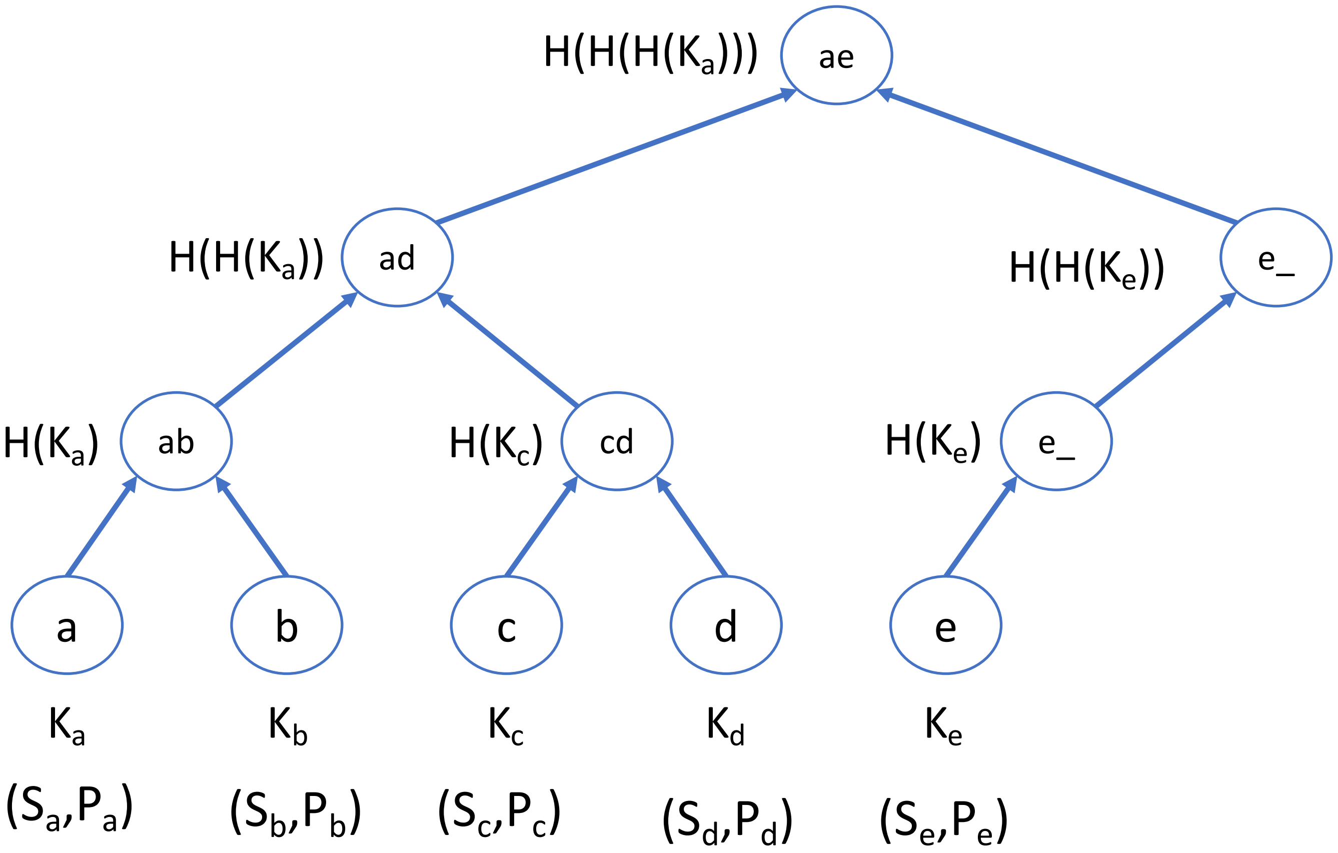
- *Send Update:* log n DH ops
- *Recv Update:* 1..log(n) DH ops

TreeKEM: mKEM with Trees

Members = {a,b,c,d,e}

$K_0 = H(H(H(K_a)))$

$K = H(K_0, K_1, \ldots.)$



Before Setup

- *Send Create:* n ENC
- *Recv Create:* 1..log(n) DEC

After Setup

- *Send Update:* log n ENC
- *Recv Update:* 1 DEC

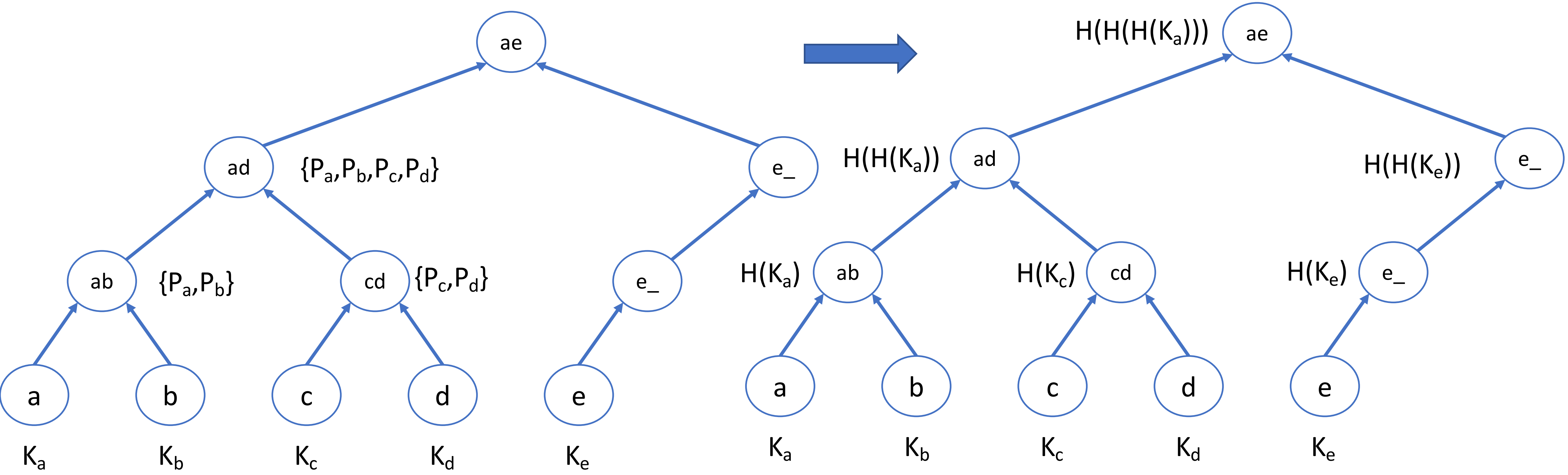
Moving between mKEM and TreeKEM

Members = {a,b,c,d,e}

Key = $\{P_a,P_b,P_c,P_d,P_e\}$

Members = {a,b,c,d,e}

Key = $\mathbf{H(H(H(K_a)))}$



TreeKEM vs. ART

PRIMITIVES

Public-key Encryption, PRF, AEAD

EFFICIENCY

log N ENC for sender,
1 DEC for receiver

CONTRIBUTIVITY

Every sender's contribution hashed into messaging group key

Only last sender's contribution hashed into subgroup keys

PRIMITIVES

DH, PRF

EFFICIENCY

log N DH for sender,
log N DH for receiver

CONTRIBUTIVITY

Every member's leaf key used to compute the messaging group key and *all subgroup keys*