Security Analysis and Improvements for the IETF MLS Standard for Group Messaging

Joël Alwen – Wickr

Sandro Coretti-Drayton – IOHK

Yevgeniy Dodis – NYU

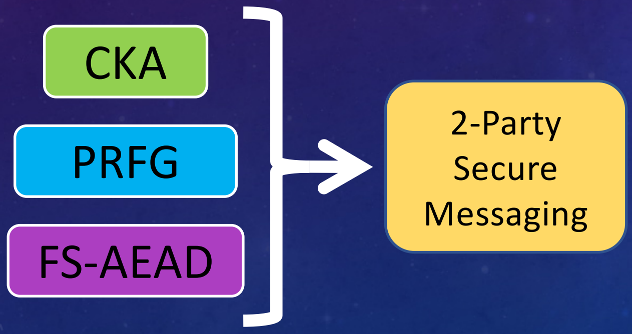
Yiannis Tselekounis – NYU

RWC 2020

COMPOSITION (FOLLOWING [ACD19])

- [ACD19]: Modularizes & generalizes (2-party) Signal's Double-Ratchet.
- The MLS Protocol: can also be viewed using a group variant of the ACD19 paradigm.

2-Party Case
(e.g., Signal)



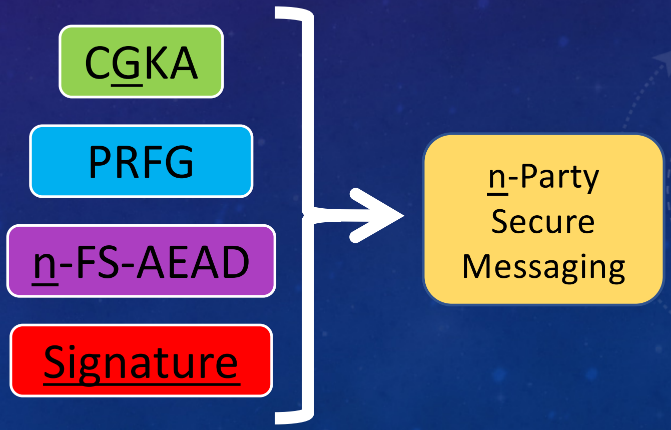
CKA

PRFG

FS-AEAD

2-Party Secure Messaging

Group Case
(e.g., MLS protocol)



CGKA

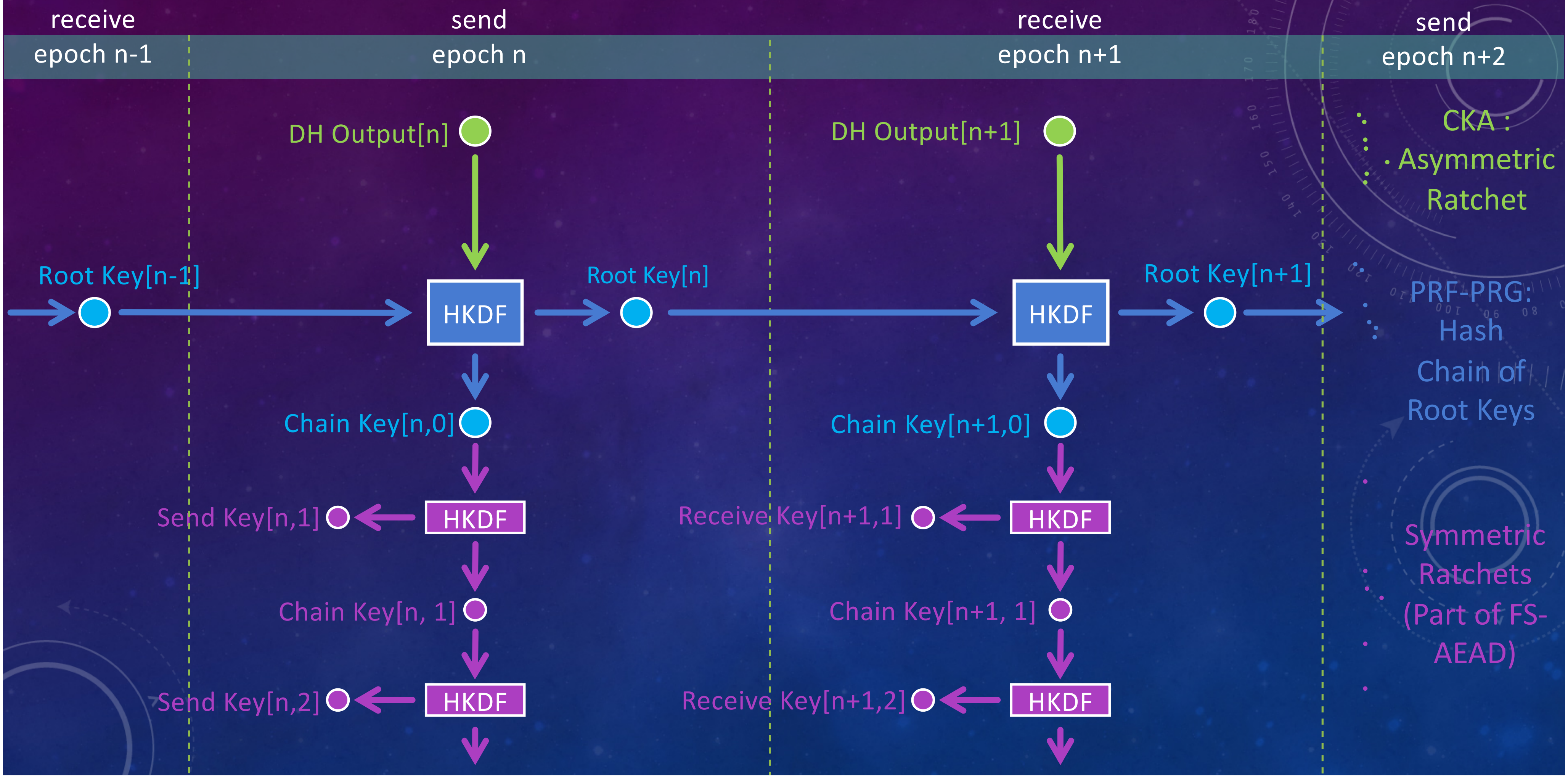
PRFG

n-FS-AEAD

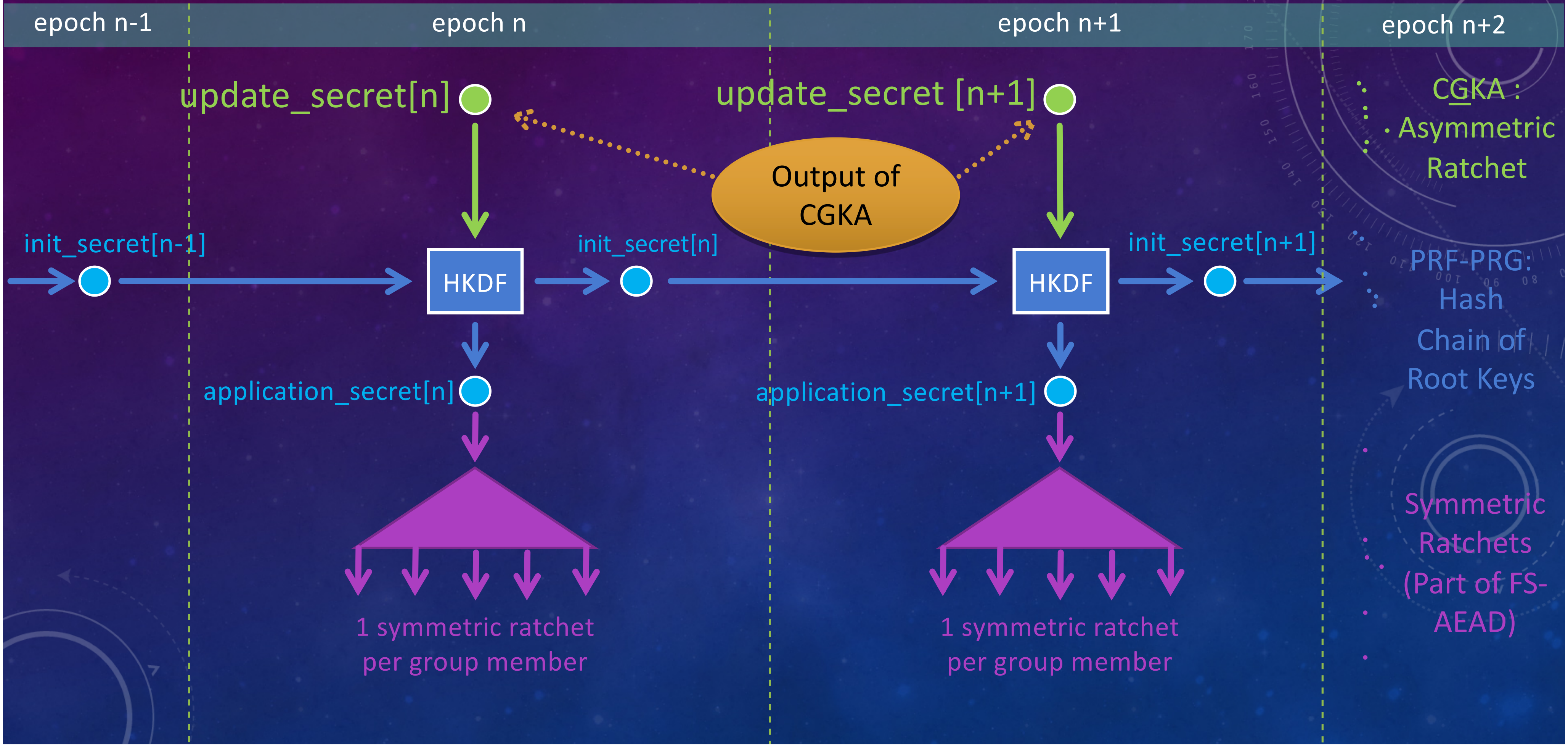
Signature

n-Party Secure Messaging

ACD19 VIEW OF DOUBLE-RATCHET



GROUP-ACD19 VIEW OF MLS



epoch n-1

epoch n

epoch n+1

epoch n+2

update_secret[n]

update_secret [n+1]

Output of CGKA

init_secret[n-1]

HKDF

init_secret[n]

HKDF

init_secret[n+1]

application_secret[n]

application_secret[n+1]

1 symmetric ratchet per group member

1 symmetric ratchet per group member

CGKA : Asymmetric Ratchet

PRF-PRG: Hash Chain of Root Keys

Symmetric Ratchets (Part of FS-AEAD)

TREEKEM: CRITICAL KEYS

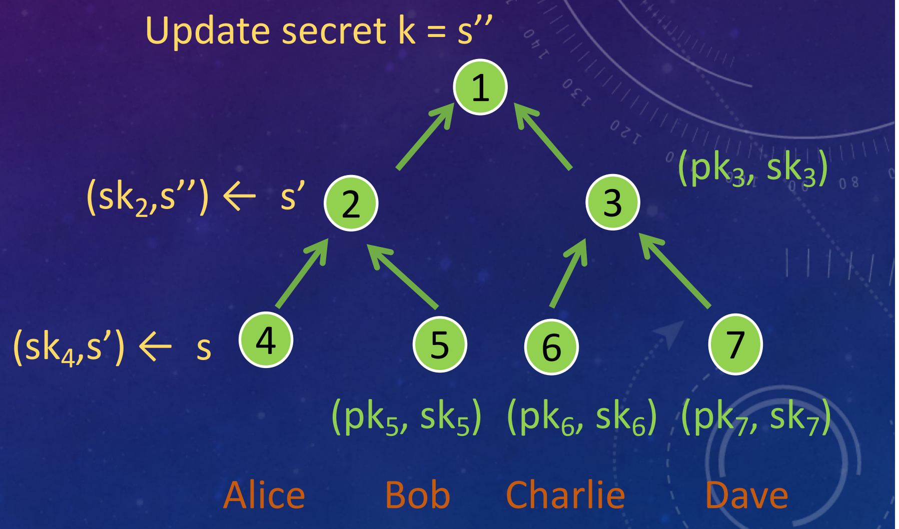
Question: When can we claim that update secret s'' is Forward Secure?

Definition: An sk is *critical for secret s* ⇔ knowing sk and all network traffic reveals s.

Observation: s'' is not FS until all critical keys for s'' removed from ratchet tree.

Our Example:

1. sk_5 is critical for s' and thus for s''.
2. sk_3 is critical for s''.



Generated ciphertexts: $c_5 \leftarrow E(pk_5,s')$

$c_3 \leftarrow E(pk_3,s'')$

TREEKEM: CRITICAL KEYS

Recursively critical keys

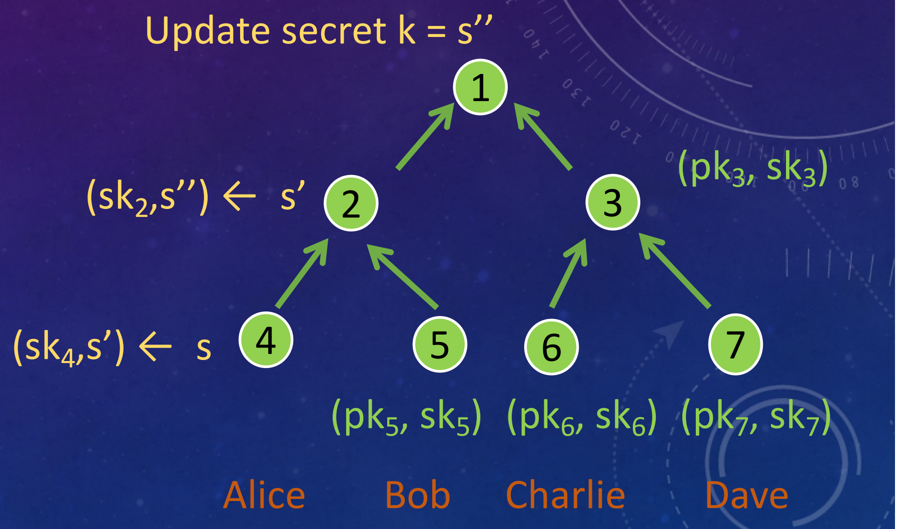
Question: When can we claim that update secret s’’ is Forward Secure?

Definition: An sk is *critical for secret s* ⇔ knowing sk and all network traffic reveals s.

Observation: s’’ is not FS until all critical keys for s’’ removed from ratchet tree.

Our Example:

1. sk_5 is critical for s’ and thus for s’’.
2. sk_3 is critical for s’’.
3. either sk_6 or sk_7 is critical for s-value from which sk_3 was generated



Update secret k = s’’

$(sk_2,s'') \leftarrow s'$

(pk_3, sk_3)

$(sk_4,s') \leftarrow s$

(pk_5, sk_5) (pk_6, sk_6) (pk_7, sk_7)

Generated ciphertexts: $c_5 \leftarrow E(pk_5,s')$

$c_3 \leftarrow E(pk_3,s'')$

TREEKEM: CRITICAL KEYS

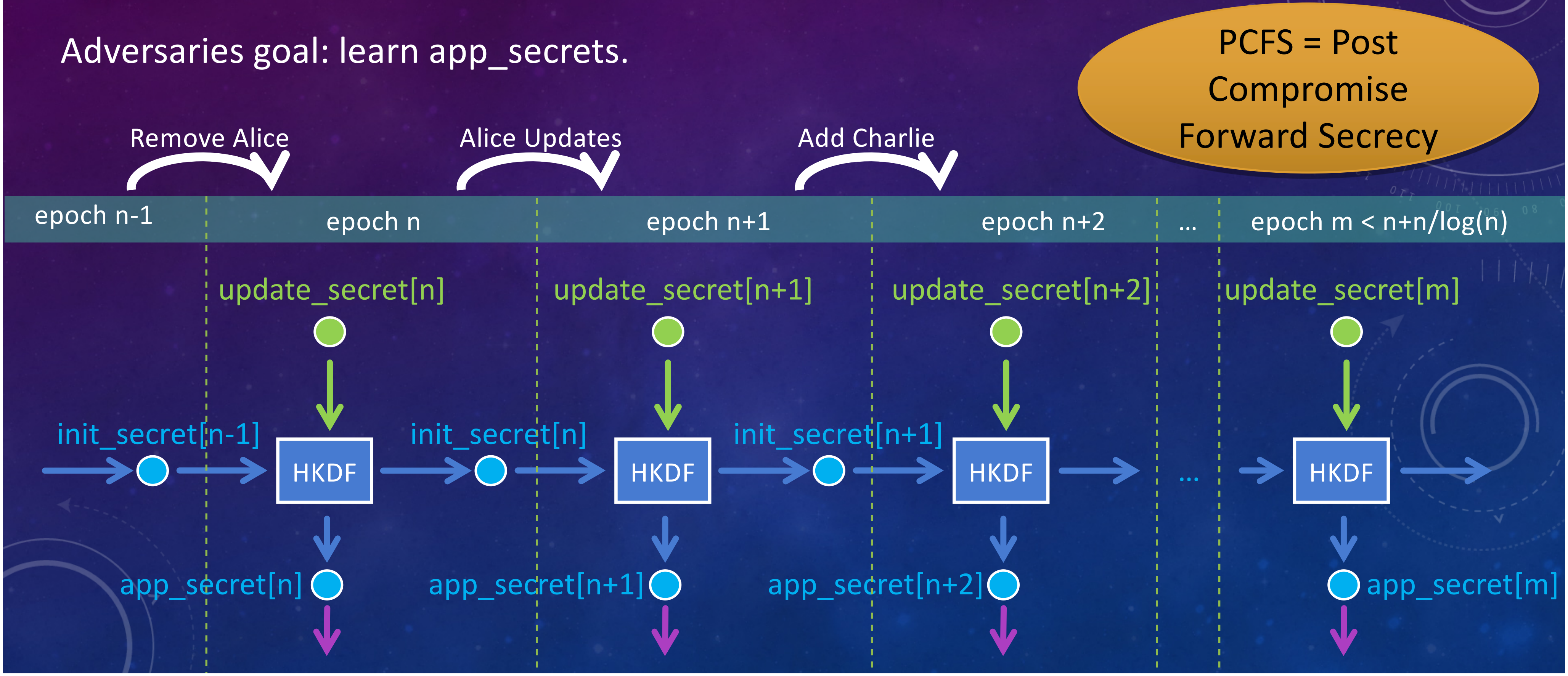
Lemma: if |G|=n, immediately following any TreeKEM update operation, the root secret generated by this update has at least n - 1 (out of 2n-1 total!) critical keys in the tree.

Why is this a problem? Because FS takes a *long* time to kick in.

- Each update overwrites at most log(n) keys => ½ n epochs to get FS even in the best case, even if nobody corrupted yet!
  - Optimal security requires FS after a single update!
- Worst case indefinite, if the right people (e.g., sibling of the updating leaf) don't perform updates!

POOR FS FOR TREEKEM POOR “PCFS” FOR MLS

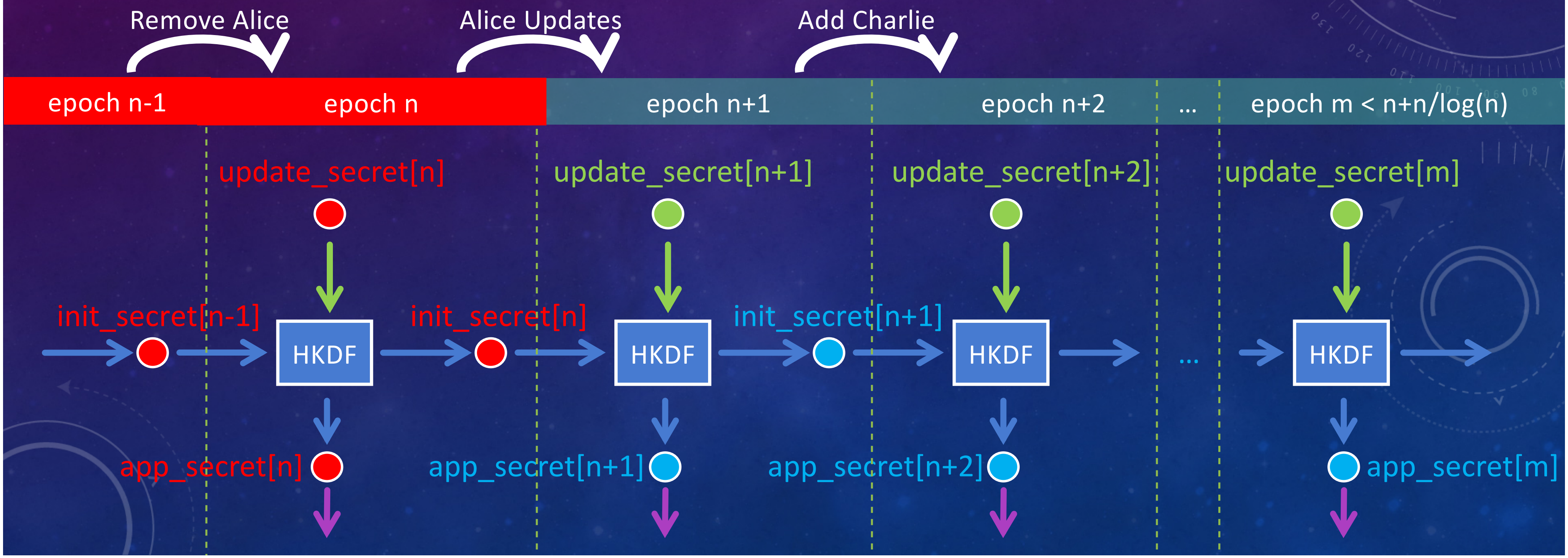
Adversaries goal: learn app_secrets.



PCFS = Post Compromise Forward Secrecy

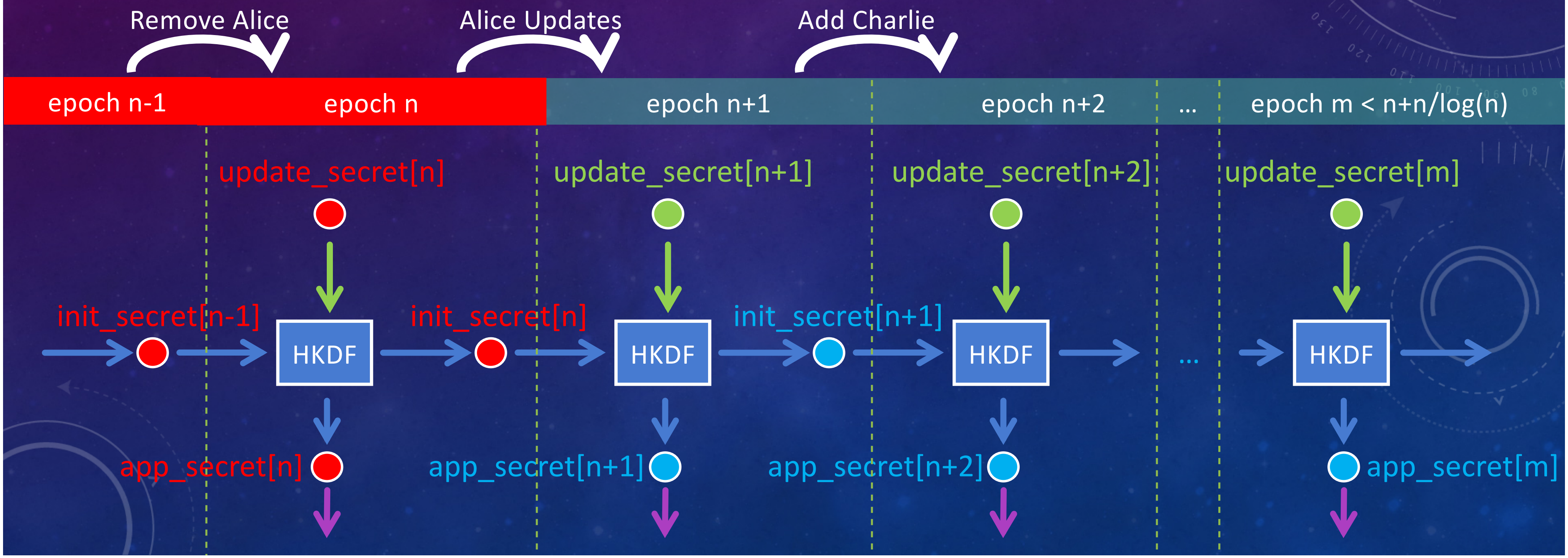
POOR FS FOR TREEKEM � POOR “PCFS” FOR MLS

Suppose adversary compromised Alice between her last update and epoch n…



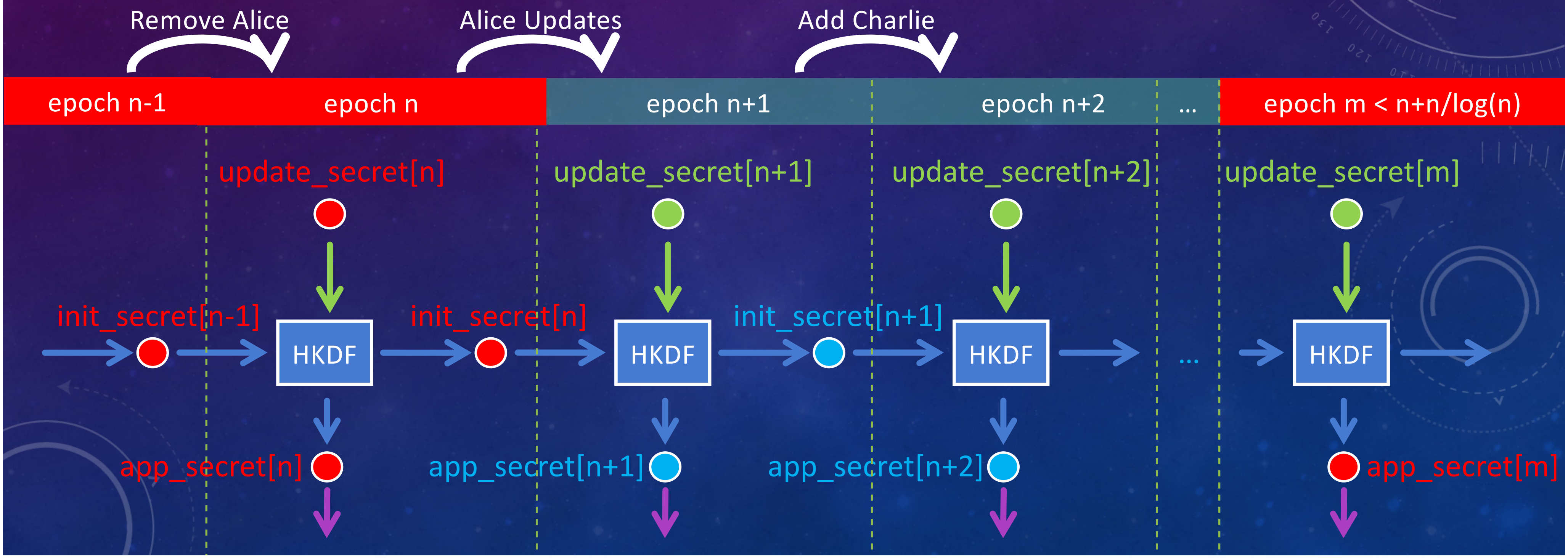
POOR FS FOR TREEKEM ⍰ POOR “PCFS” FOR MLS

Epoch n : Alice updates. Adversary cant decrypt. So is app_secret[n+1] FS when group reaches epoch n+2?



POOR FS FOR TREEKEM ⇒ POOR “PCFS” FOR MLS

Adversary corrupts Dave during epoch n+3. Can’t invert HKDF so



Remove Alice

Alice Updates

Add Charlie

epoch n-1

epoch n

epoch n+1

epoch n+2

…

epoch m < n+n/log(n)

update_secret[n]

update_secret[n+1]

update_secret[n+2]

update_secret[m]

init_secret[n-1]

init_secret[n]

init_secret[n+1]

HKDF

HKDF

HKDF

…

HKDF

app_secret[n]

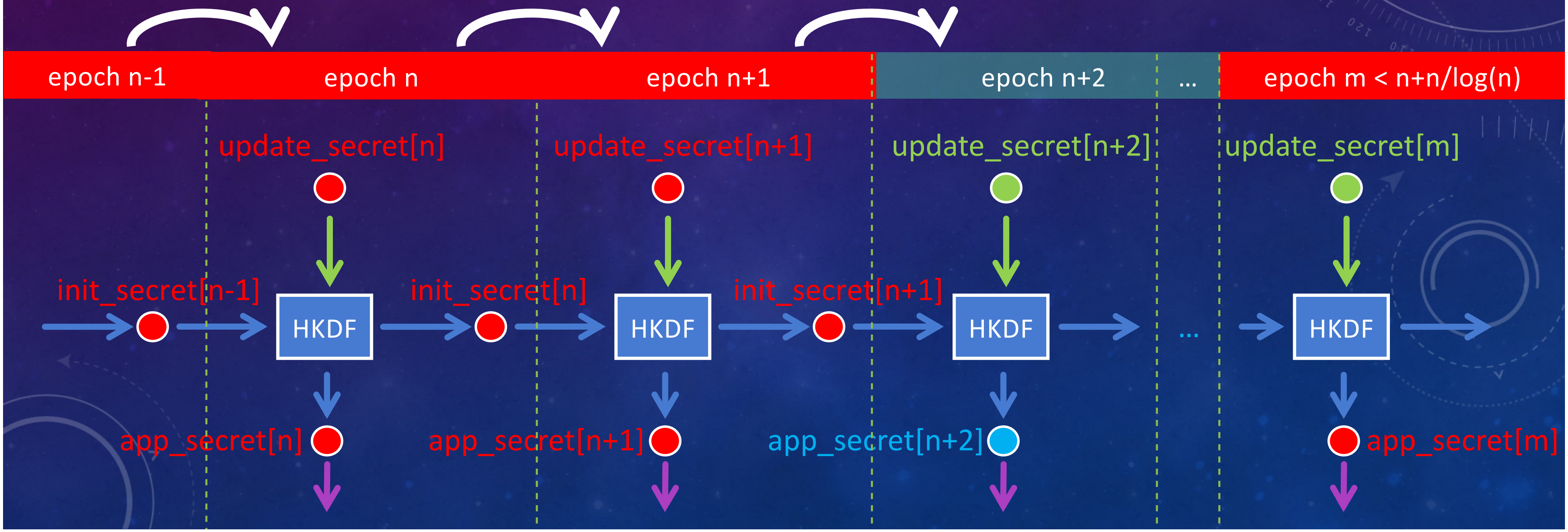
app_secret[n+1]

app_secret[n+2]

app_secret[m]

POOR FS FOR TREEKEM ⍰ POOR “PCFS” FOR MLS

…but Dave had critical key k for update_secret[n+1]!



epoch n-1

epoch n

epoch n+1

epoch n+2

…

epoch m < n+n/log(n)

update_secret[n]

update_secret[n+1]

update_secret[n+2]

update_secret[m]

init_secret[n-1]

init_secret[n]

init_secret[n+1]

HKDF

HKDF

HKDF

…

HKDF

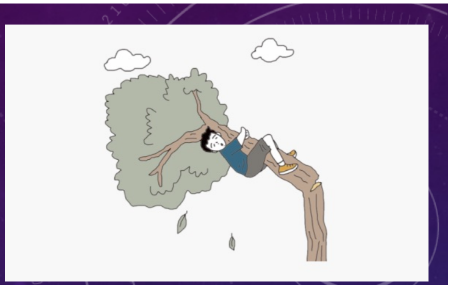
app_secret[n]

app_secret[n+1]

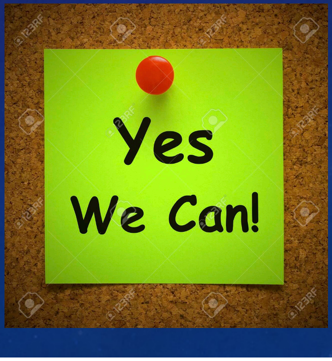
app_secret[n+2]

app_secret[m]

INSECURITY OF TREEKEM



- Lemma: TreeKEM achieves *less-than-ideal* FS, even under the most favorable circumstances
- In the paper we characterize *precisely* the set of secure keys given a sequence of attacker’s queries
  - Polynomial time computable, but complex and unintuitive (graph reachability on “key graph”)
  - Very far from optimal security
- Can we do better? Optimal?



Replacing standard PKE in TreeKEM with “Updatable PKE” yields an optimally secure CGKA protocol (called RTreeKEM).

- Closely related to “Key-Updateable PKE” used for 2-party secure messaging protocol in [JMM @ Eurocrypt’18]
- Inspired by proposal of Konrad Kohbrok. [MLS mailing list 24/Jan/2019]
- Intuition: Practical Forward Secure PKE

STANDARD PKE

- Syntax:

  $(pk, sk) \leftarrow \mathrm{KeyGen}(1^\lambda)$

  $c \leftarrow \mathrm{Enc}(pk, m)$

  $m \leftarrow \mathrm{Dec}(sk, m)$

- Correctness: senders need not be synchronized

UPDATABLE PKE

- Syntax:

  $(pk_0, sk_0) \leftarrow \mathrm{KeyGen}(1^\lambda)$

  $(\ c_i\ ,\ pk_i\ ) \leftarrow \mathrm{Enc}(pk_{i-1}, m_i)$

  $(\ m_i\ ,\ sk_i\ ) \leftarrow \mathrm{Dec}(sk_{i-1}, c_i)$

- Correctness: only if all senders are “synchronized”
  - OK by MLS assumption!

STANDARD (ELGAMAL) PKE

random

- **KG:** $pk \leftarrow g^{sk}$
- **Enc of** m: $c \leftarrow (g^r, H(pk^r) \oplus m)$
- **Dec of** (c_1,c_2): $m \leftarrow H(c_1^{sk}) \oplus c_2$

(ADDITIVE) UPDATABLE PKE

- **KG:** $pk \leftarrow g^{sk}$
- **Enc of** m: $d' \leftarrow \{0,1\}^{256}$

  $d = HKDF(d', context)$

  $c \leftarrow (g^r, H(pk^r) \oplus (m \| d'))$

  $pk \leftarrow pk \cdot g^d$
- **Dec of** (c_1,c_2): $(m \| d') \leftarrow H(c_1^{sk}) \oplus c_2$

  $sk \leftarrow sk + HKDF(d', context)$

STANDARD (ELGAMAL) PKE

random

- **KG:** $pk \leftarrow g^{sk}$
- **Enc of** m: $c \leftarrow (g^r, H(pk^r) \oplus m)$
- **Dec of** (c_1,c_2): $m \leftarrow H(c_1^{sk}) \oplus c_2$

(MULTIPLICATIVE) UPDATABLE PKE

- **KG:** $pk \leftarrow g^{sk}$
- **Enc of** m: $d' \leftarrow \{0,1\}^{256}$

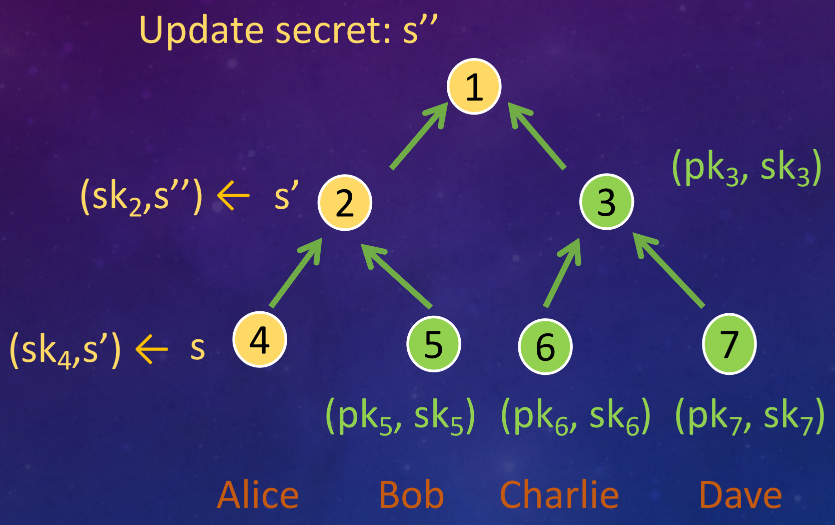
  $d = HKDF(d', context)$

  $c \leftarrow (g^r, H(pk^r) \oplus (m \,\|\, d'))$

  $pk \leftarrow pk^d$
- **Dec of** (c_1,c_2): $(m \,\|\, d') \leftarrow H(c_1^{sk}) \oplus c_2$

  $sk \leftarrow sk * HKDF(d', context)$

TREEKEM AND CRITICAL KEYS



Update secret: s’’

1

$(sk_2,s’’) \leftarrow s’$ 2

3 (pk_3, sk_3)

$(sk_4,s’) \leftarrow s$ 4

5 6 7

(pk_5, sk_5) (pk_6, sk_6) (pk_7, sk_7)

Alice Bob Charlie Dave

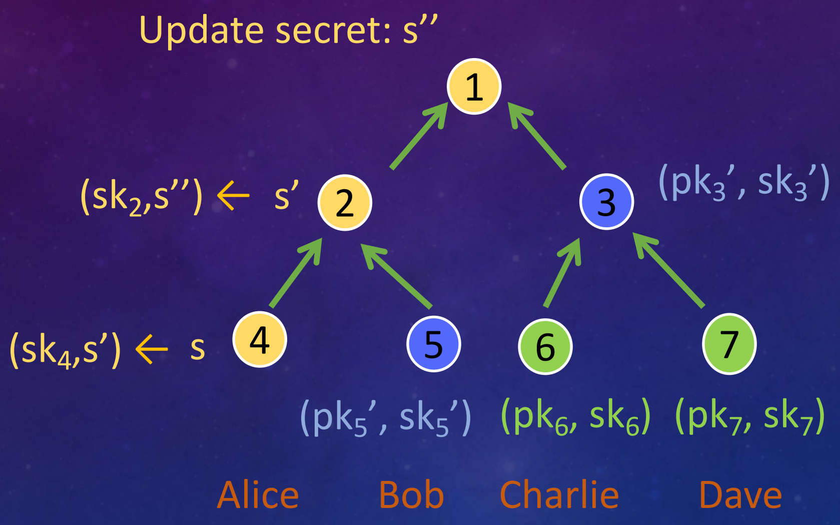
BEFORE

Generated ciphertexts: $c_5 \leftarrow E(pk_5,s’)$

$c_3 \leftarrow E(pk_3,s’’)$

RTREEKEM AND CRITICAL KEYS

AFTER



Update secret: s''

$(sk_2, s'') \leftarrow s'$ 2

$(sk_4, s') \leftarrow s$ 4

1

3 (pk_3', sk_3')

5 (pk_5', sk_5')

6 (pk_6, sk_6)

7 (pk_7, sk_7)

Alice Bob Charlie Dave

sk_5' and sk_3' now useless for update secret s''

Generated ciphertexts and new key pairs:

$(pk_5', c_5) \leftarrow E(pk_5, s')$ $(sk_5', s') \leftarrow D(sk_5, c_5)$

$(pk_3', c_3) \leftarrow E(pk_3, s'')$ $(sk_3', s'') \leftarrow D(sk_3, c_3)$

MORE RESULTS

- More results in paper [eprint/2019/1189]:
  - Security against adaptive adversary.
  - Future directions & open problems for E2E secure group messaging.
- Follow up work: (Multiplicative-)UPKE for X25519/X448
  - See: Alwen on MLS mailing list Dec/2019
  - See: draft-barnes-cfrg-mult-for-7748-00 [ABC19]

"Sometimes it's just good to sit back and get a different perspective."