Tighter provable security for TreeKEM

Karen Azari[1], **Andreas Ellison**[2]

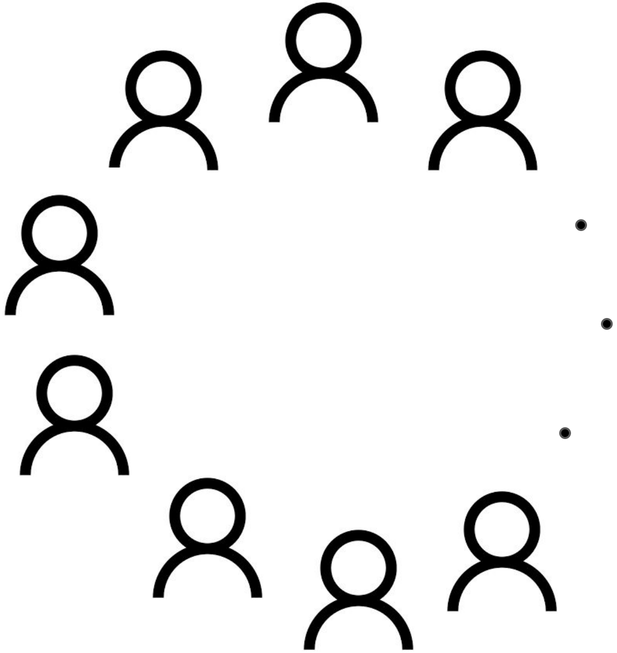
[1] universität wien

[2] ETH zürich

Outline

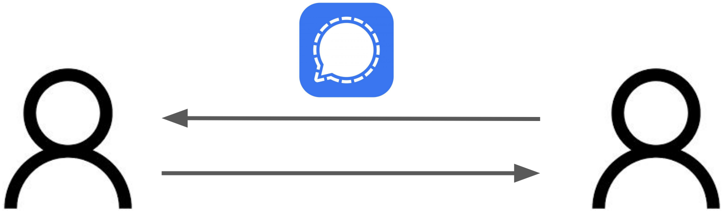
1. Big picture
2. TreeKEM
3. Our results

Big picture: End-to-end encrypted messaging in large groups

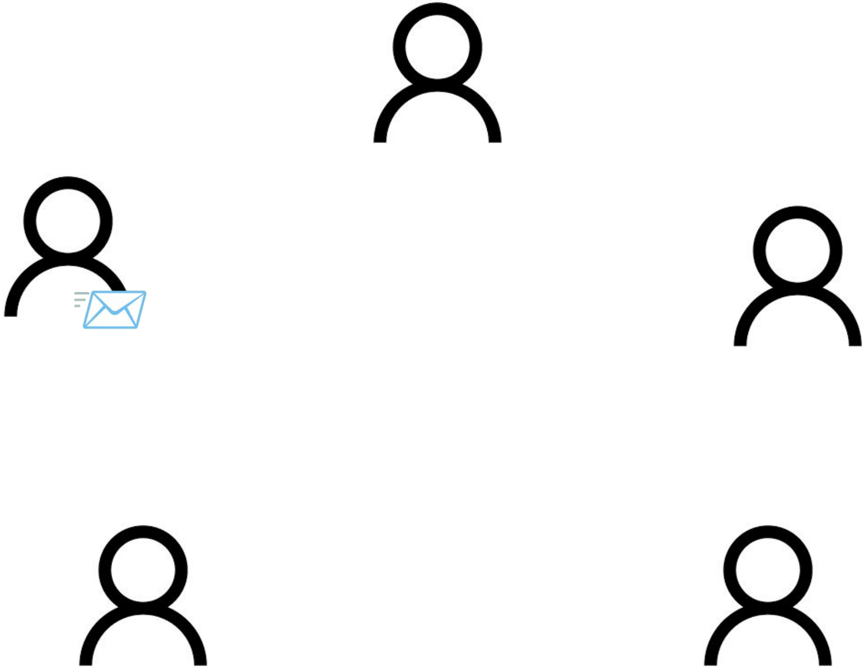


1-to-1 messaging

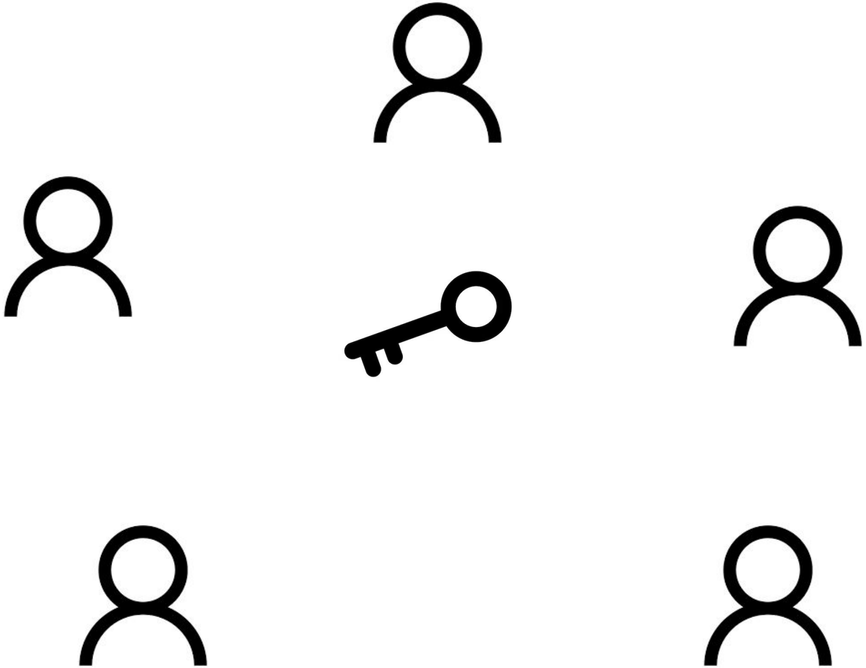
We already have secure end-to-end encrypted messaging…



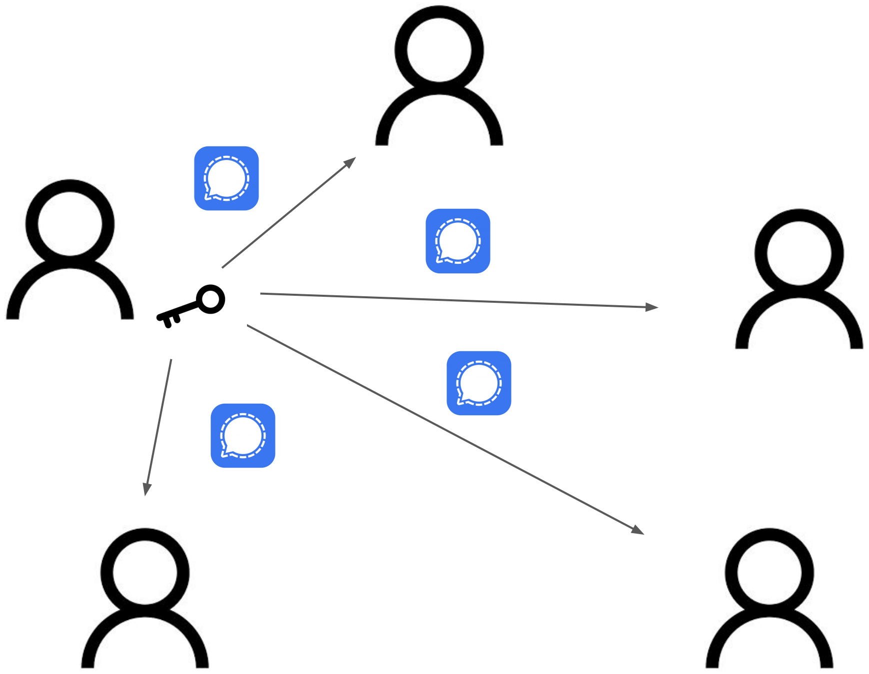
Group messaging



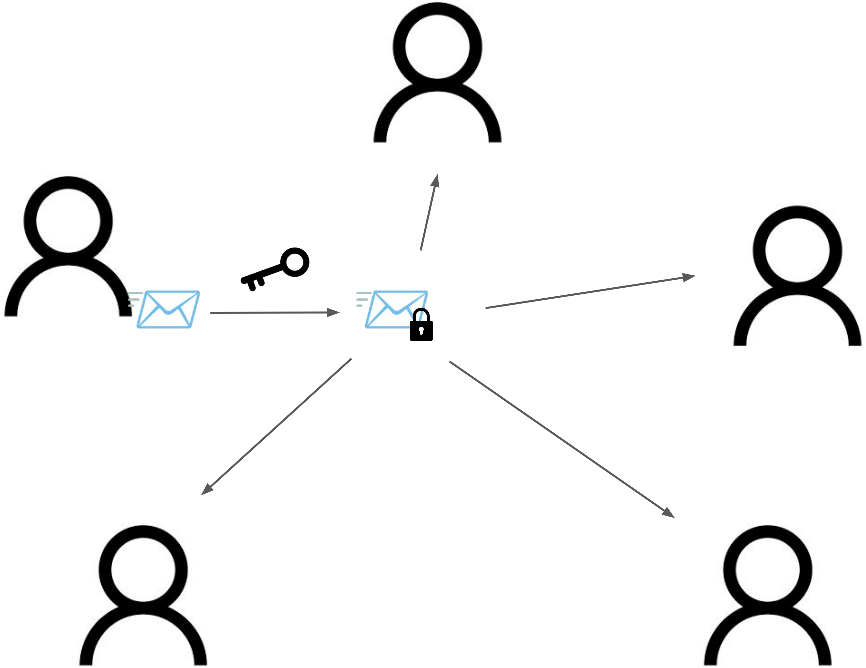
Group messaging



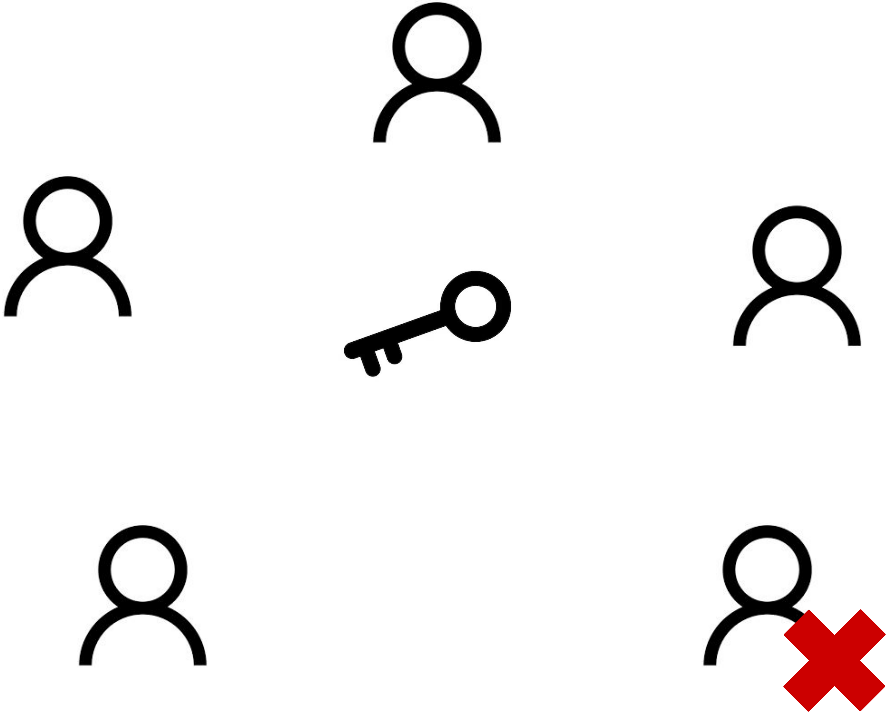
Group messaging



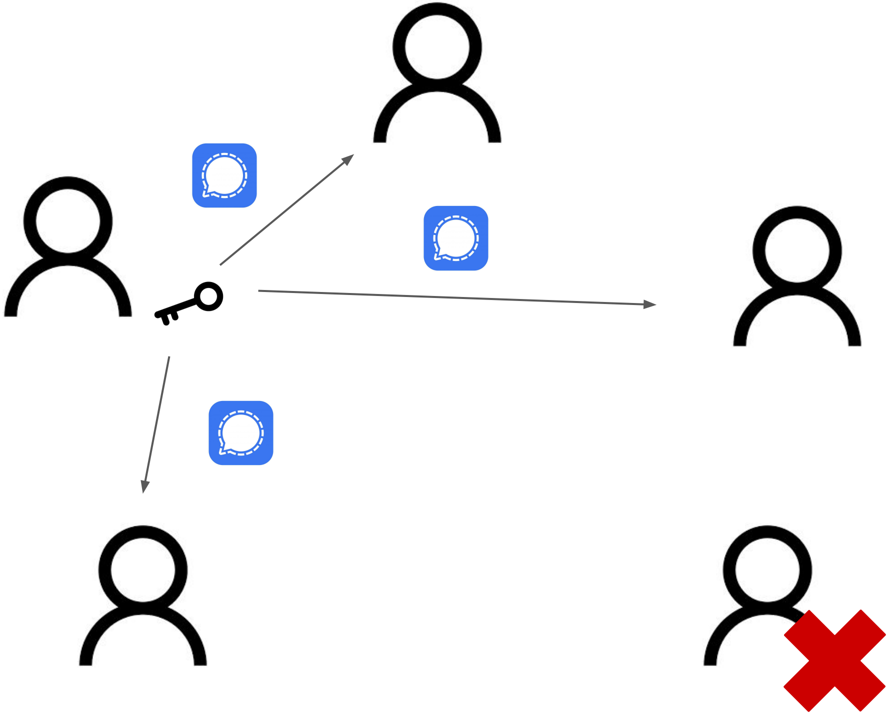
Group messaging



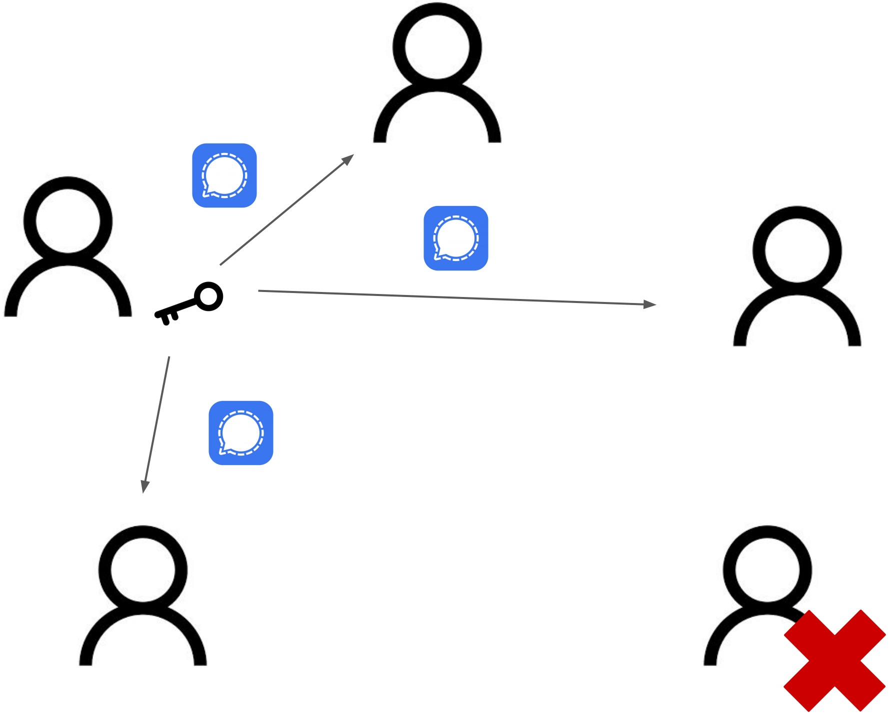
Group messaging: removing a user



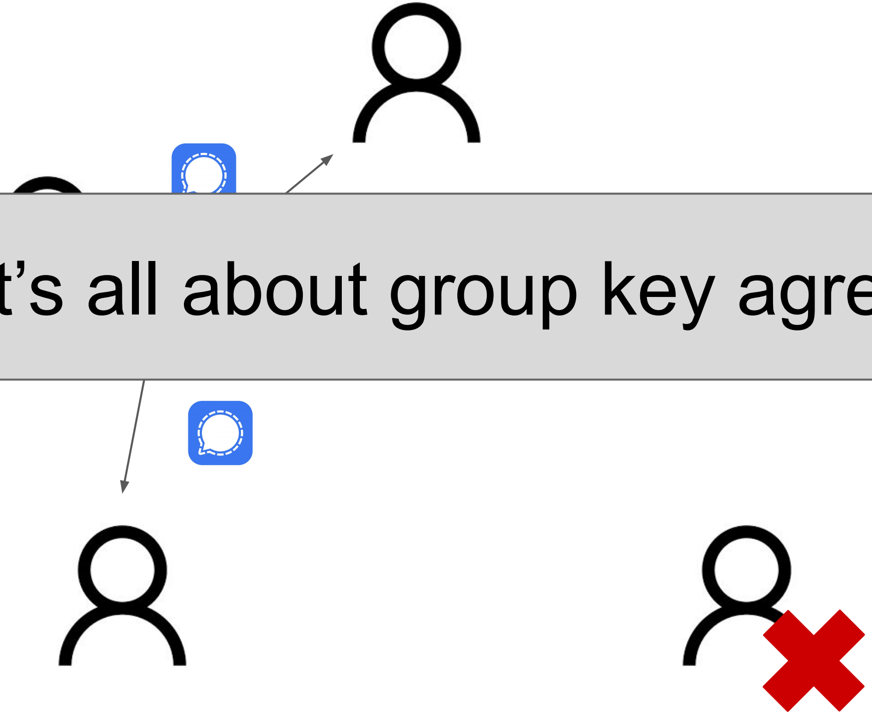
Group messaging: distribute a new key



Group messaging: distribute a new key 🐌



Group messaging: distribute a new key 🐌



💡 It's all about group key agreement!

Enter MLS…

RFC 9420
The Messaging Layer Security (MLS) Protocol

Abstract

Messaging applications are increasingly making use of end-to-end security mechanisms to ensure that messages are only accessible to the communicating endpoints, and not to any servers involved in delivering messages. Establishing keys to provide such protections is challenging for group chat settings, in which more than two clients need to agree on a key but may not be online at the same time. In this document, we specify a key establishment protocol that provides efficient asynchronous group key establishment with forward secrecy (FS) and post-compromise security (PCS) for groups in size ranging from two to thousands.

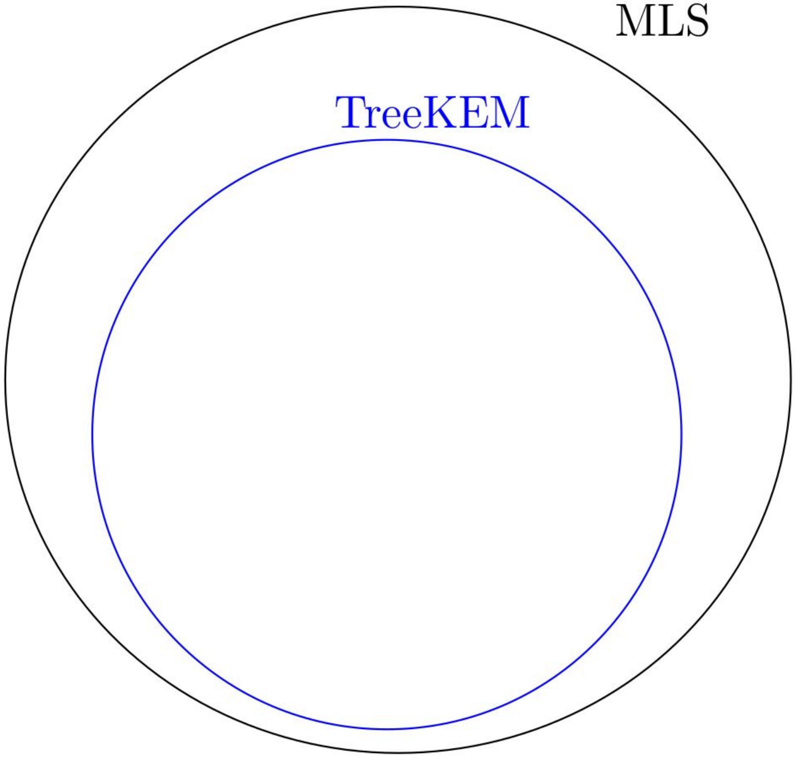
Status of This Memo

This is an Internet Standards Track document.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Further information on Internet Standards is available in Section 2 of RFC 7841.

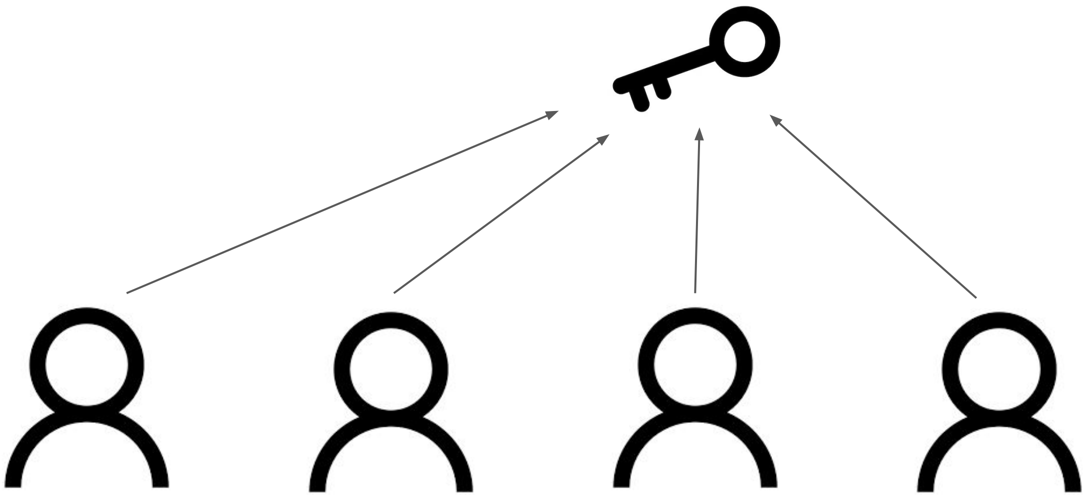
Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at https://www.rfc-editor.org/info/rfc9420.

TreeKEM provides efficient group key agreement

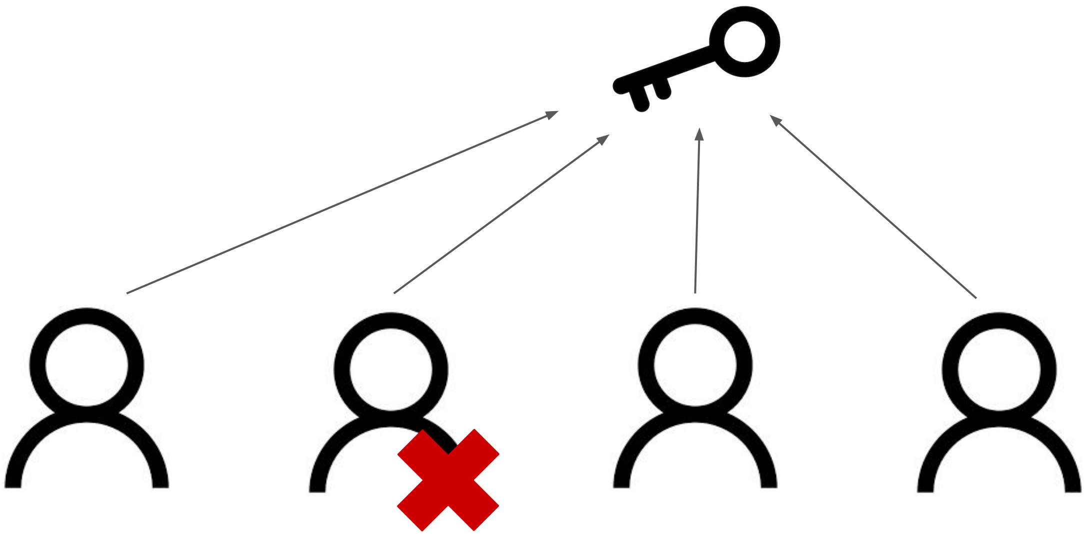


TreeKEM interface

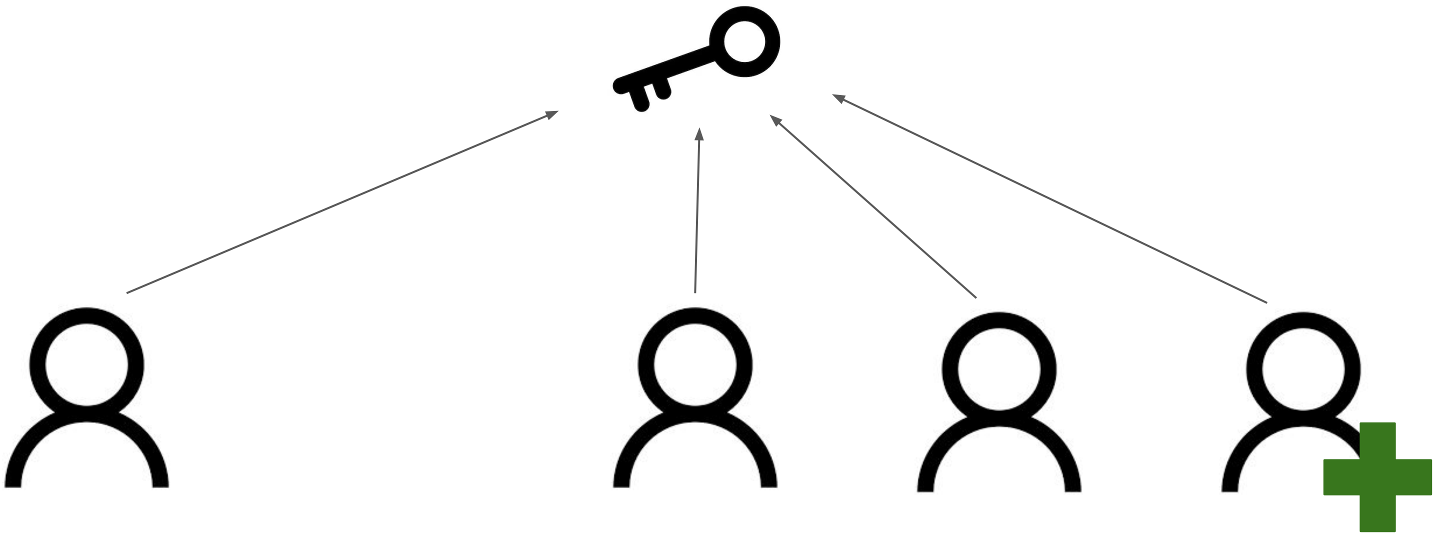
TreeKEM interface – Key agreement



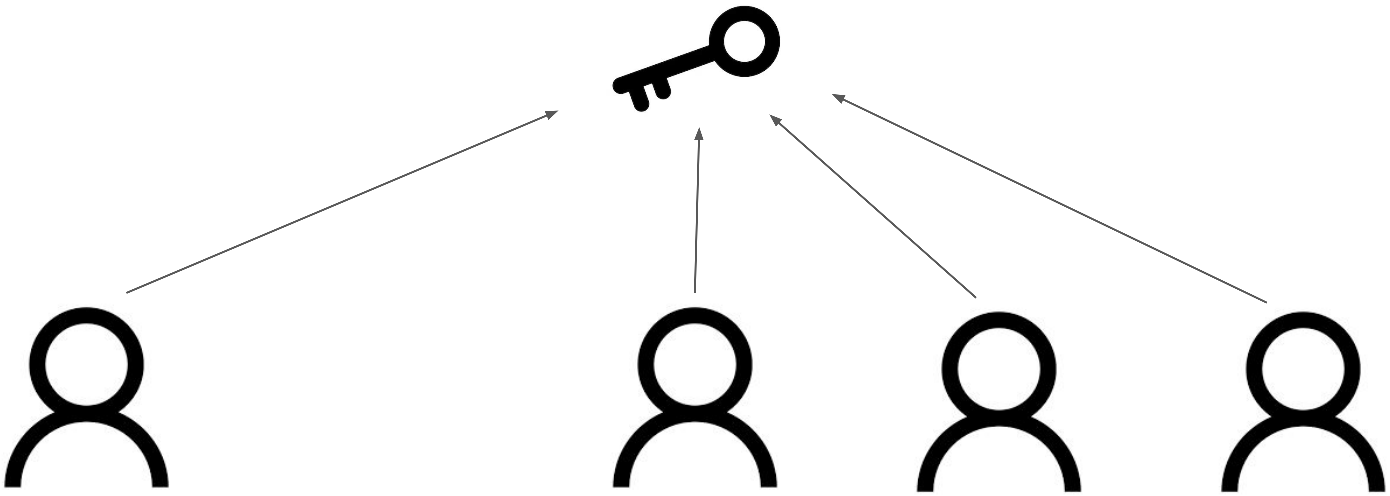
TreeKEM interface – Remove user



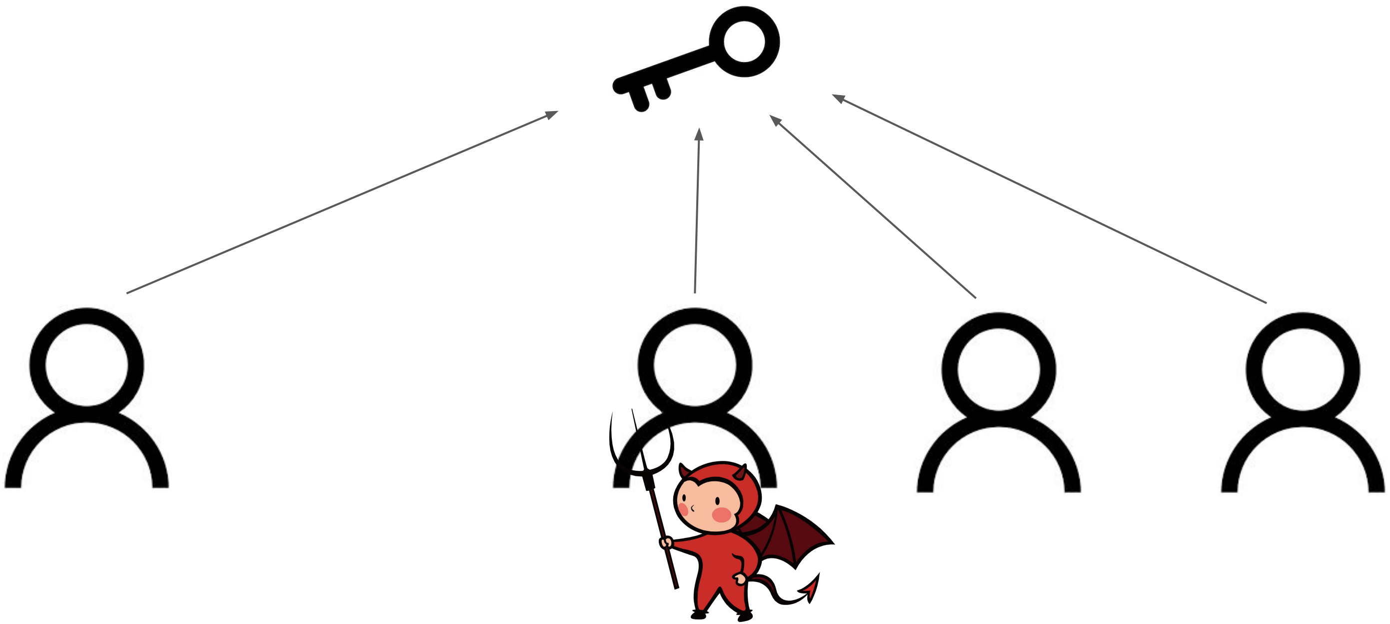
TreeKEM interface – Add user



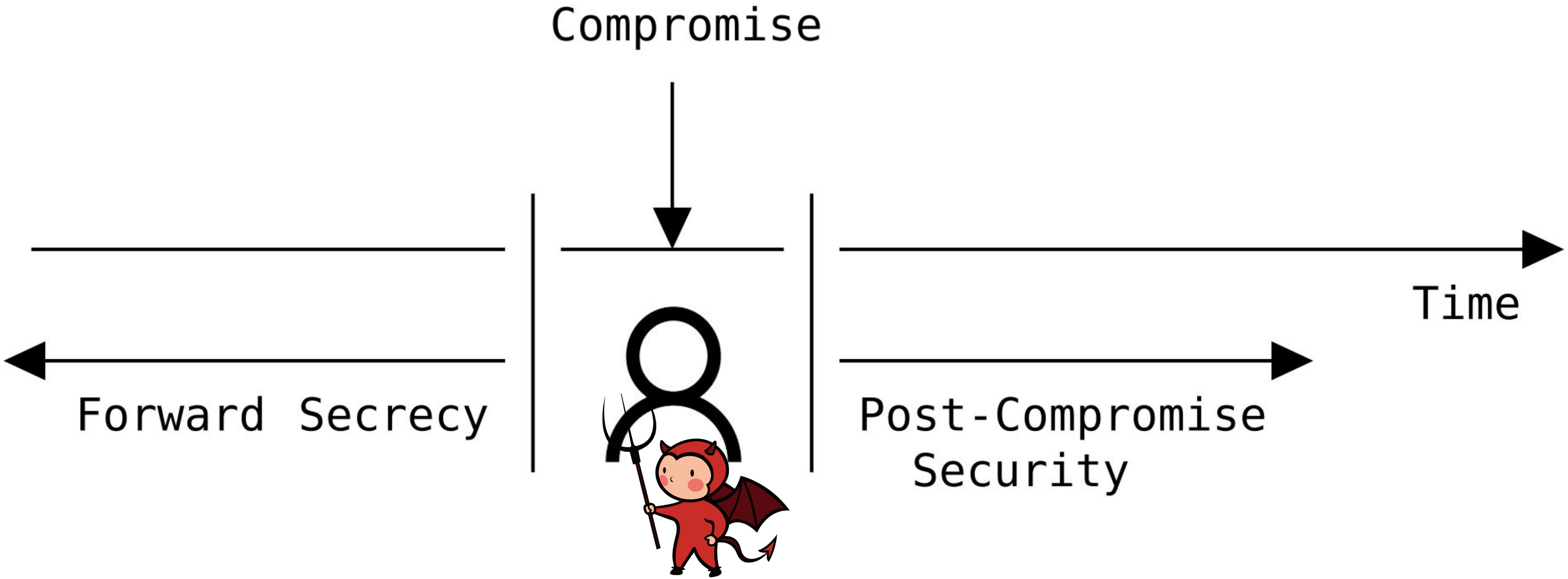
TreeKEM interface



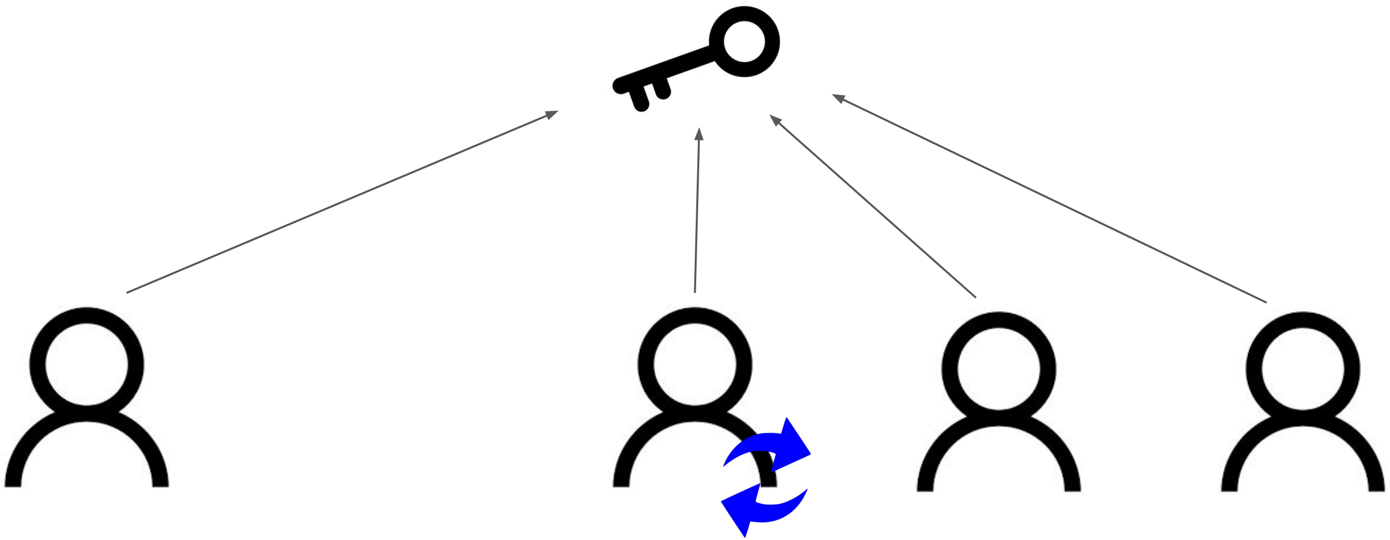
TreeKEM interface – Dealing with compromises



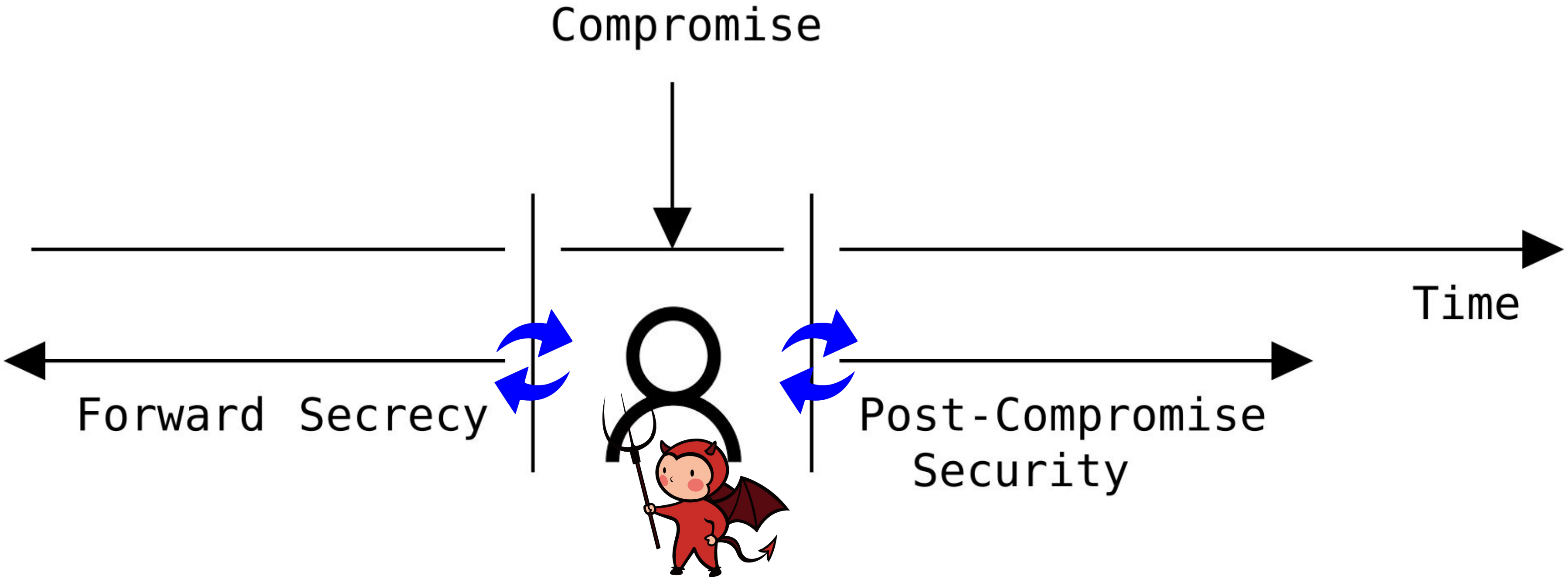
TreeKEM interface – Dealing with compromises



TreeKEM interface – Dealing with compromises

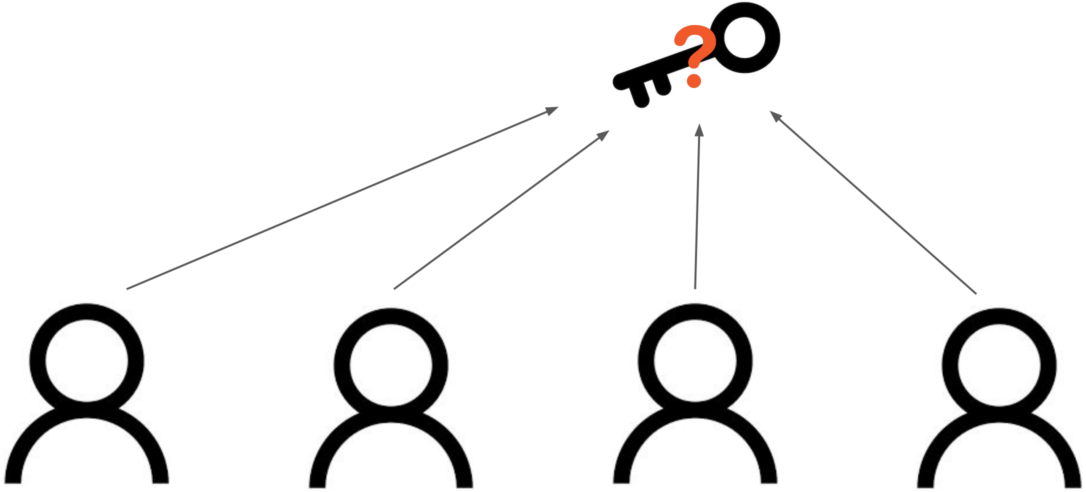


TreeKEM interface – Dealing with compromises



TreeKEM security

TreeKEM security



TreeKEM security

$\mathcal{C}$

$\xrightarrow{\text{commit}(\cdot)}$

$\xleftarrow{c_1}$

$\vdots$

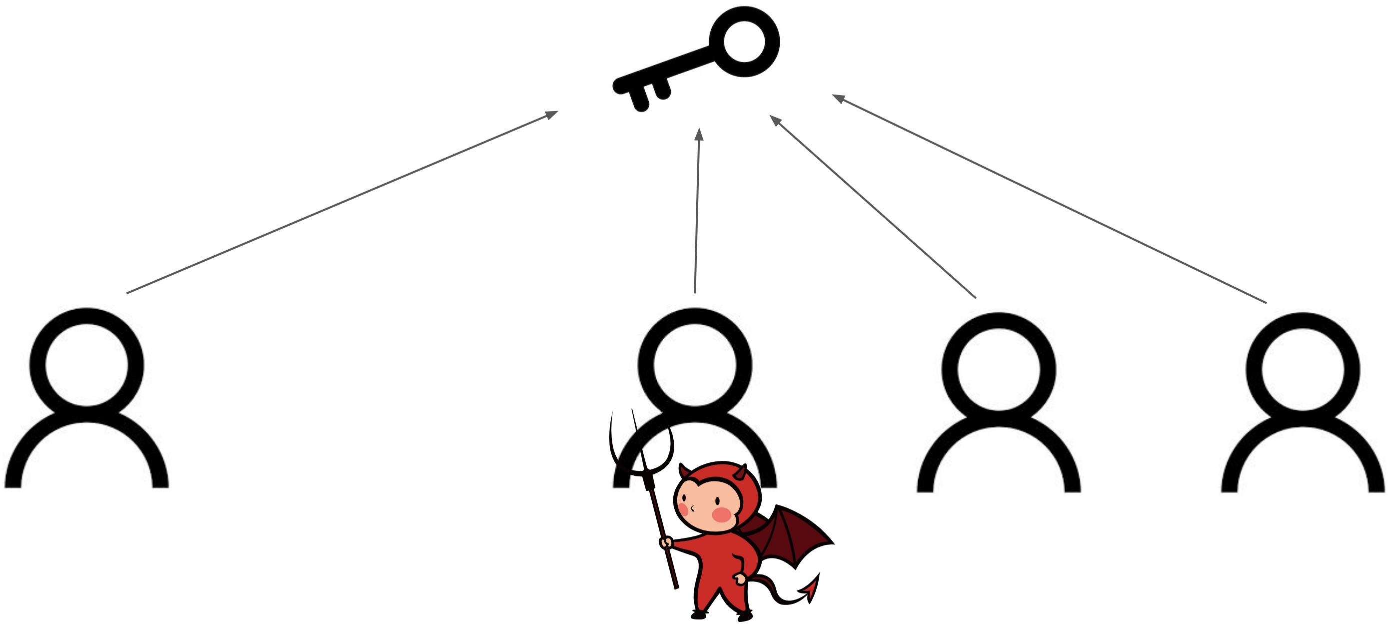
$\xrightarrow{\text{commit}(\cdot)}$

$\xleftarrow{c_n}$

$\xrightarrow{\text{challenge}(c_i)}$

$\xleftarrow{\tilde{k}}$

TreeKEM security



TreeKEM security

$\mathcal{C}$

$$\xrightarrow{\text{commit}(\cdot)}$$

$$\xleftarrow{c_1}$$

$$\vdots$$

$$\xrightarrow{\text{commit}(\cdot)}$$

$$\xleftarrow{c_n}$$

$$\xrightarrow{\text{challenge}(c_i)}$$

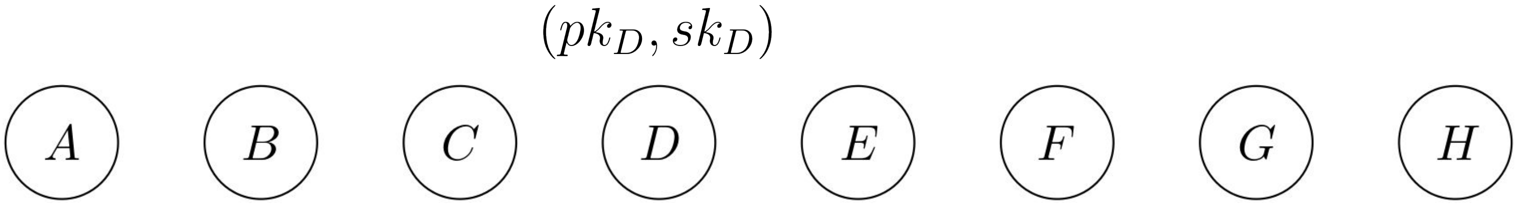
not compromised

$$\xleftarrow{\tilde{k}}$$

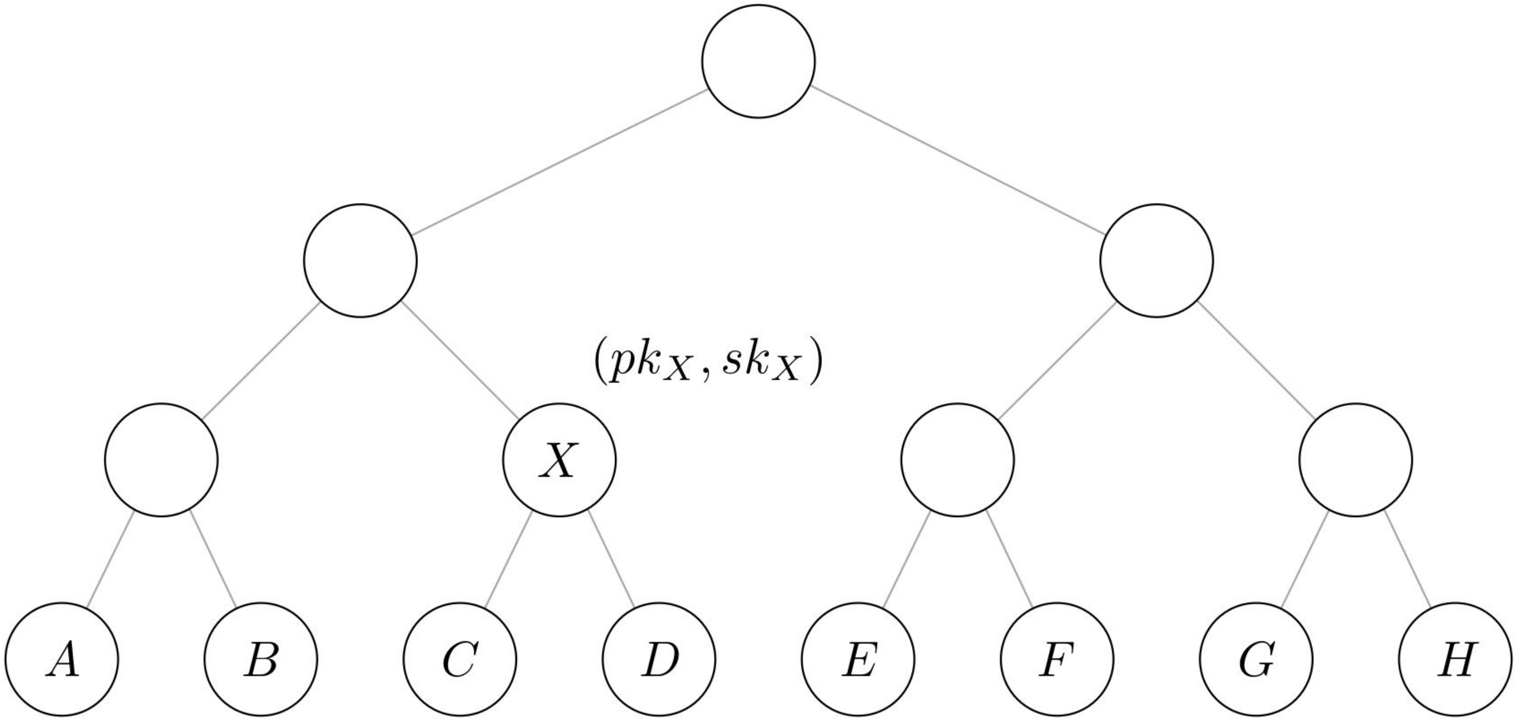
TreeKEM protocol

TreeKEM protocol

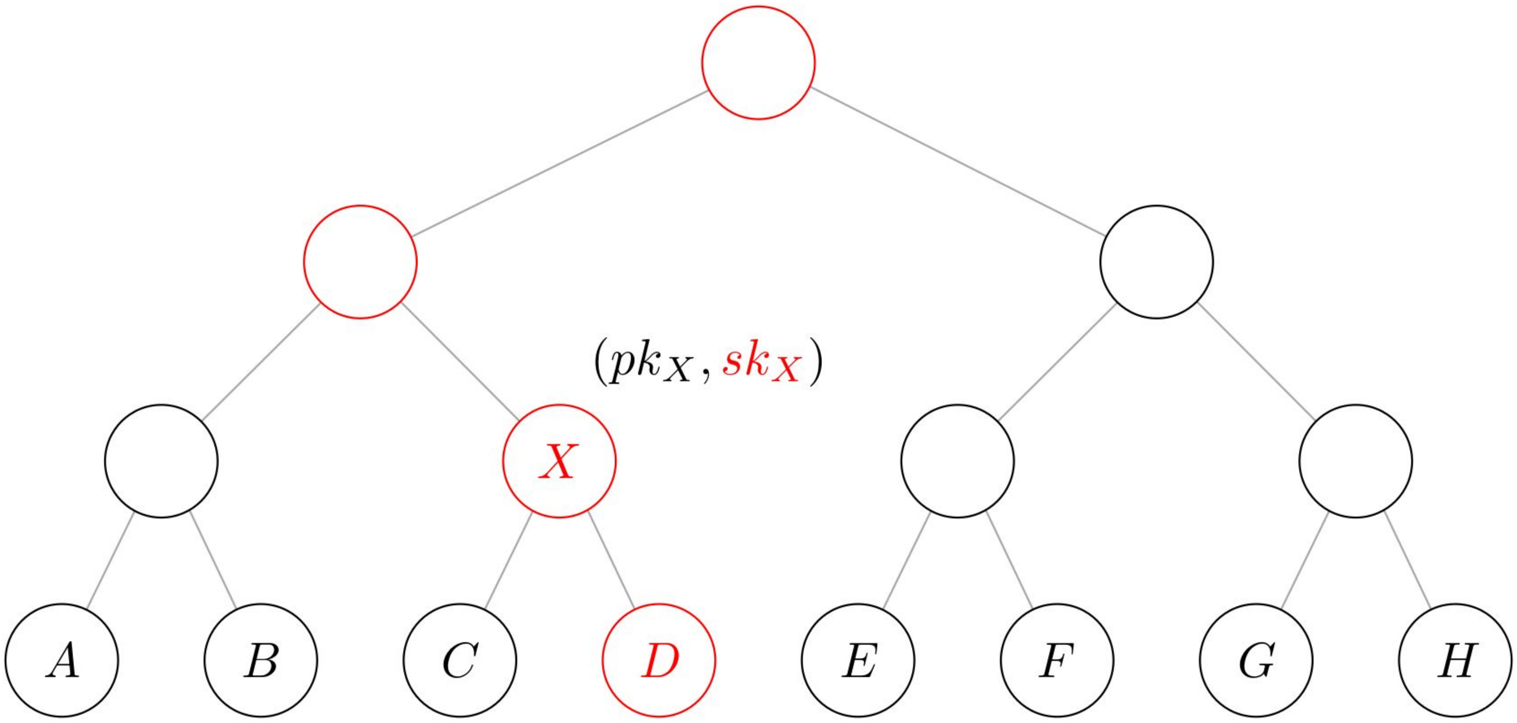
choose PKE



TreeKEM protocol

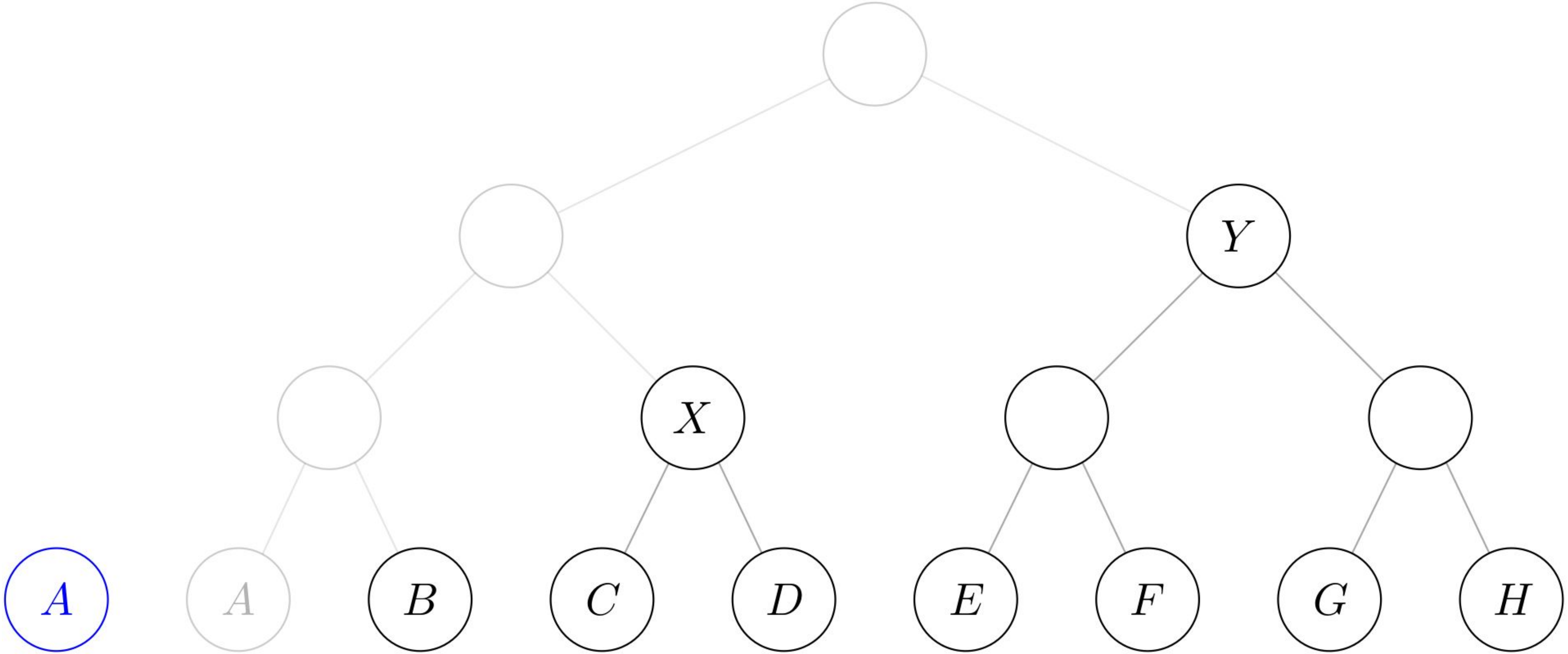


TreeKEM protocol



TreeKEM commit

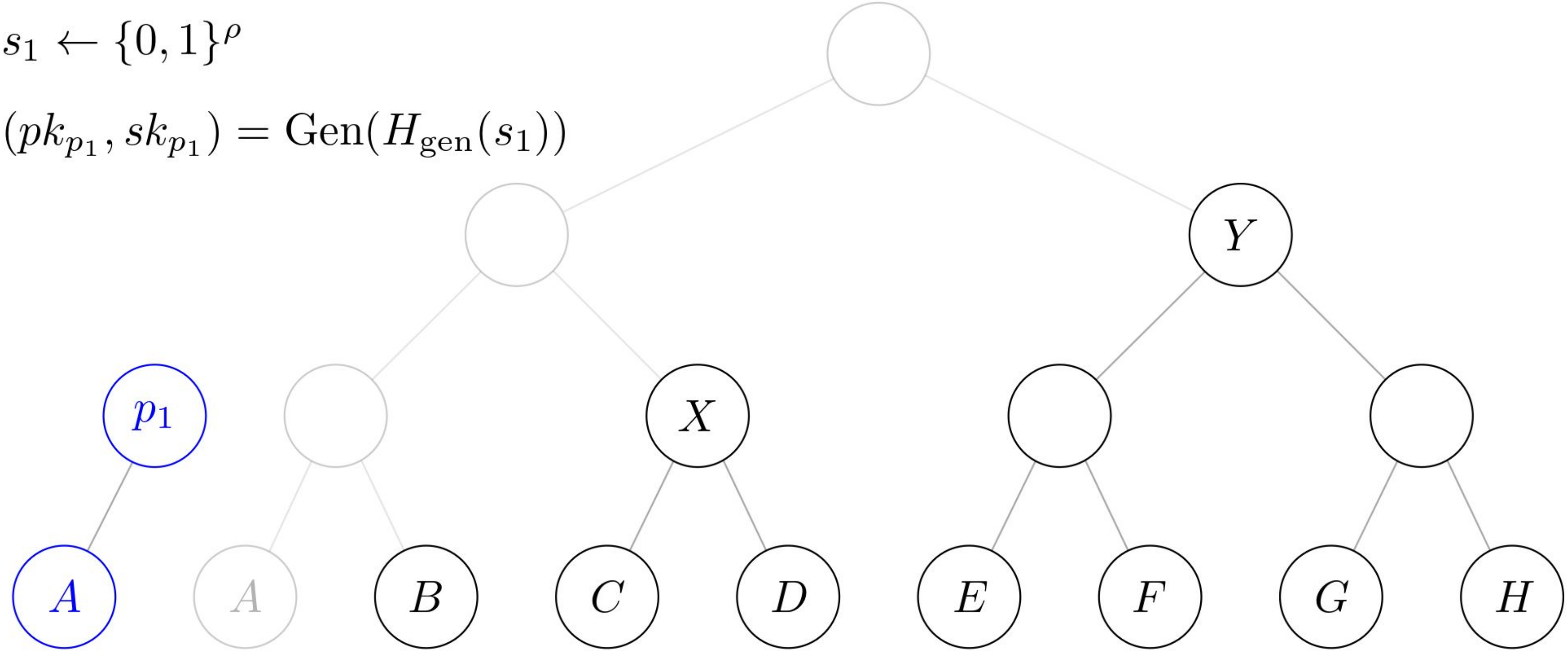
TreeKEM commit



TreeKEM commit

$$s_1 \leftarrow \{0,1\}^\rho$$

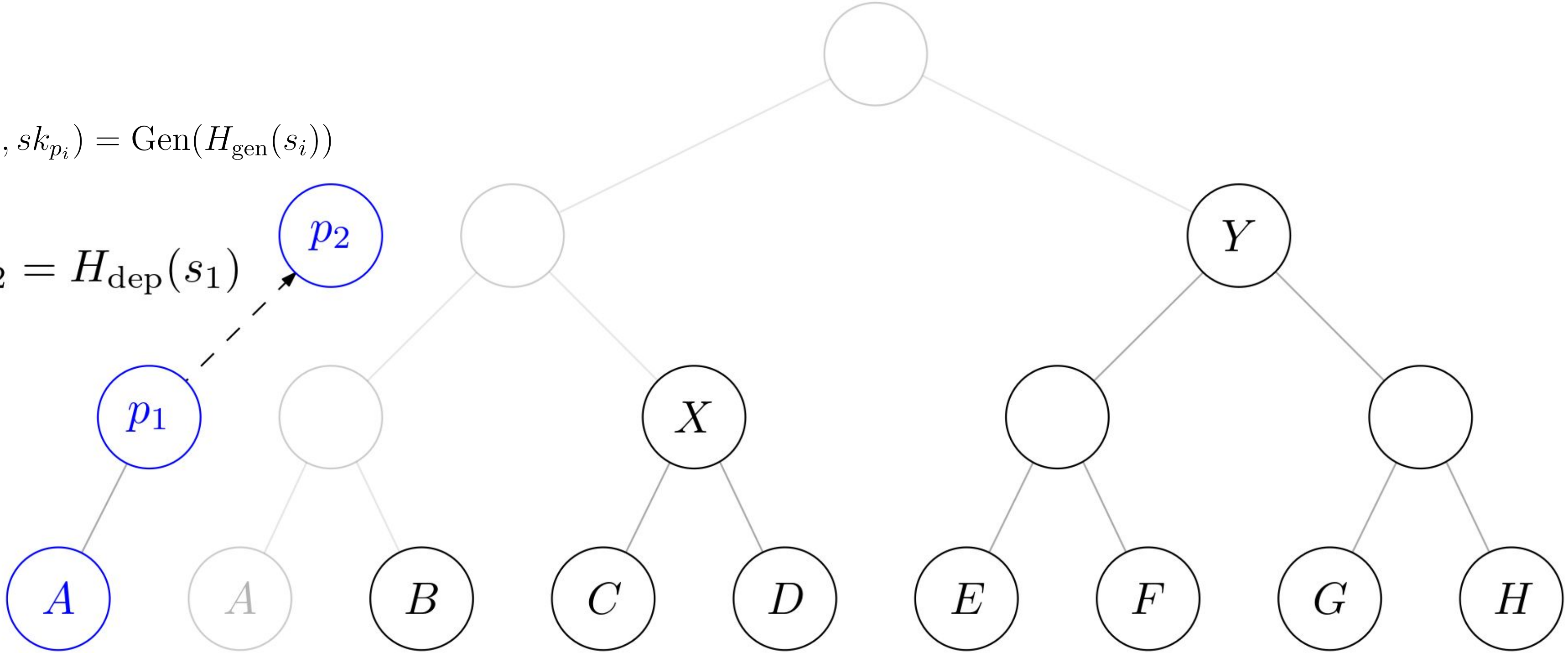
$$(pk_{p_1}, sk_{p_1}) = \text{Gen}(H_{\text{gen}}(s_1))$$



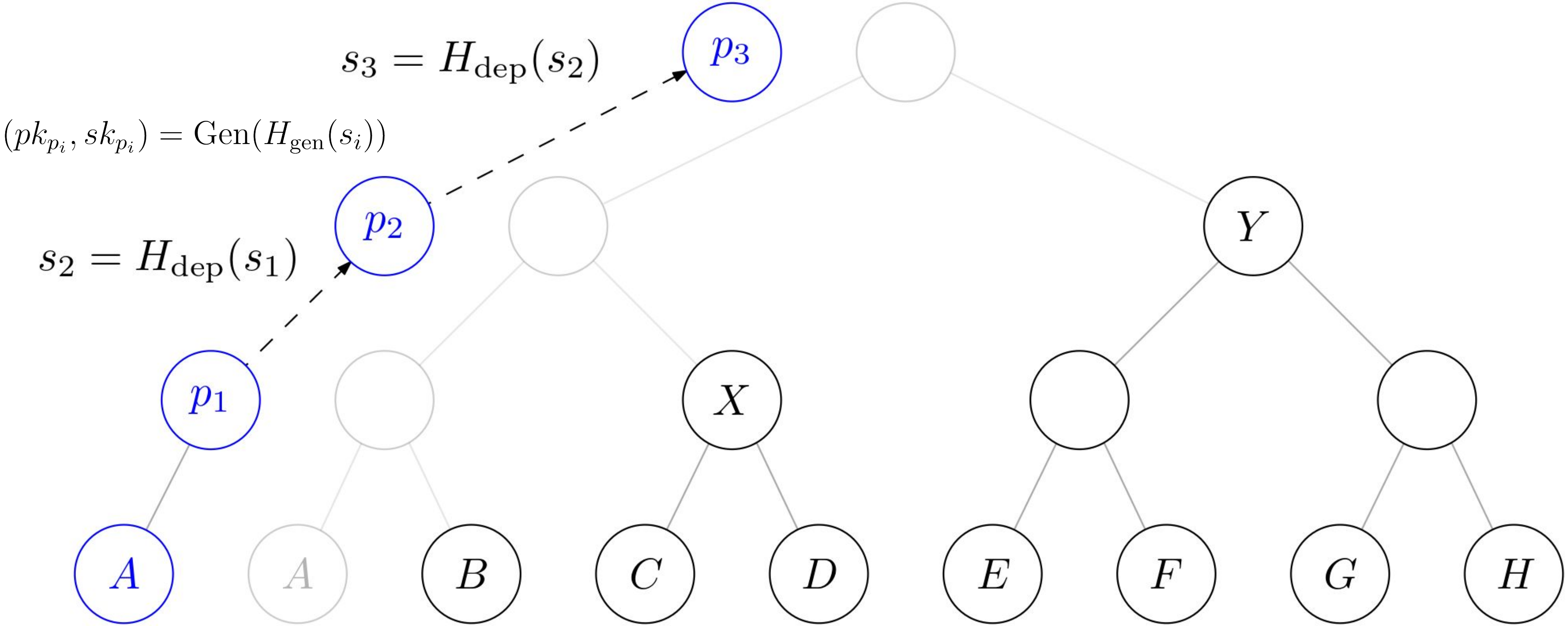
TreeKEM commit

$$(pk_{p_i}, sk_{p_i}) = \mathrm{Gen}(H_{\mathrm{gen}}(s_i))$$

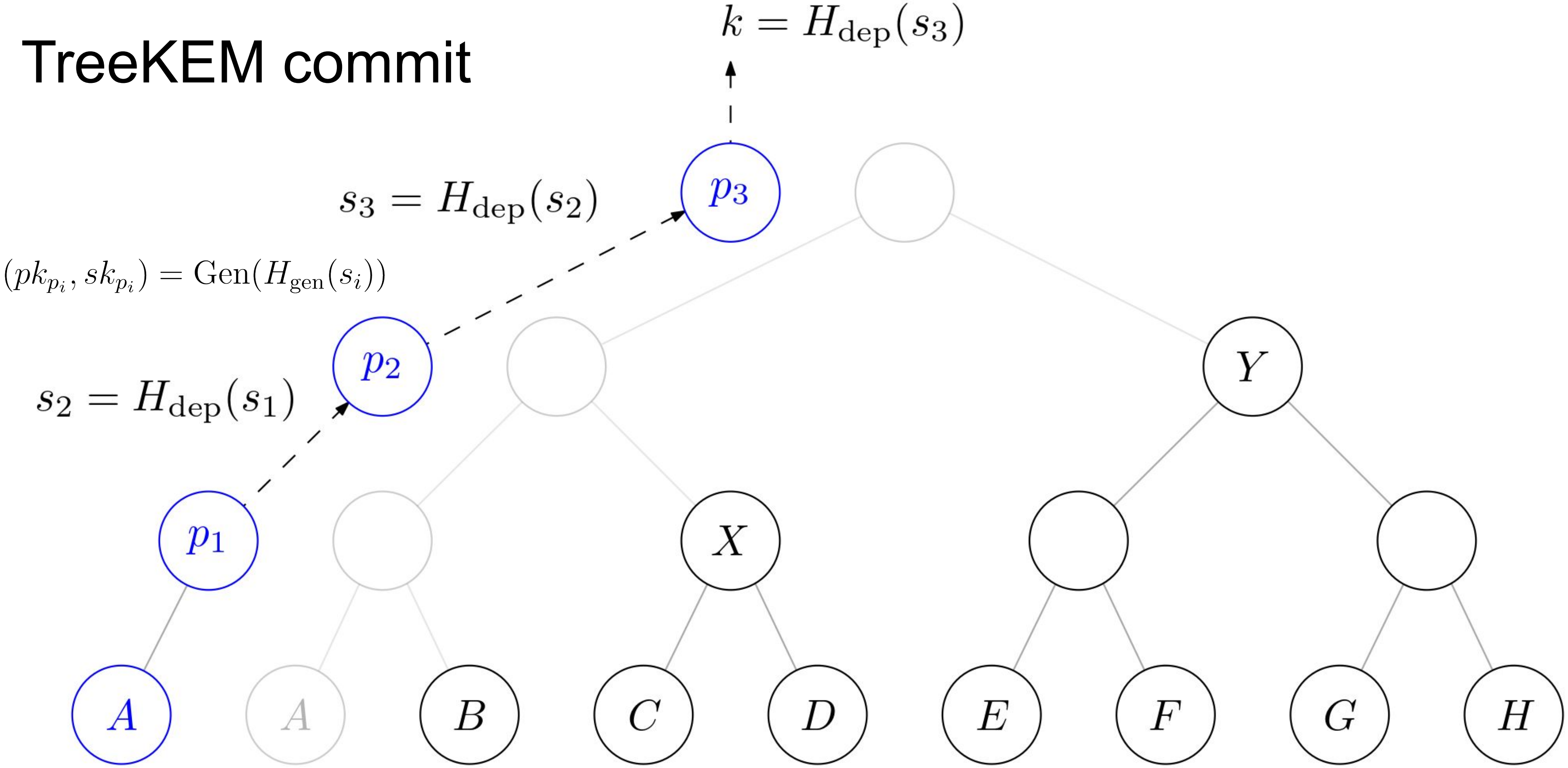
$$s_2 = H_{\mathrm{dep}}(s_1)$$



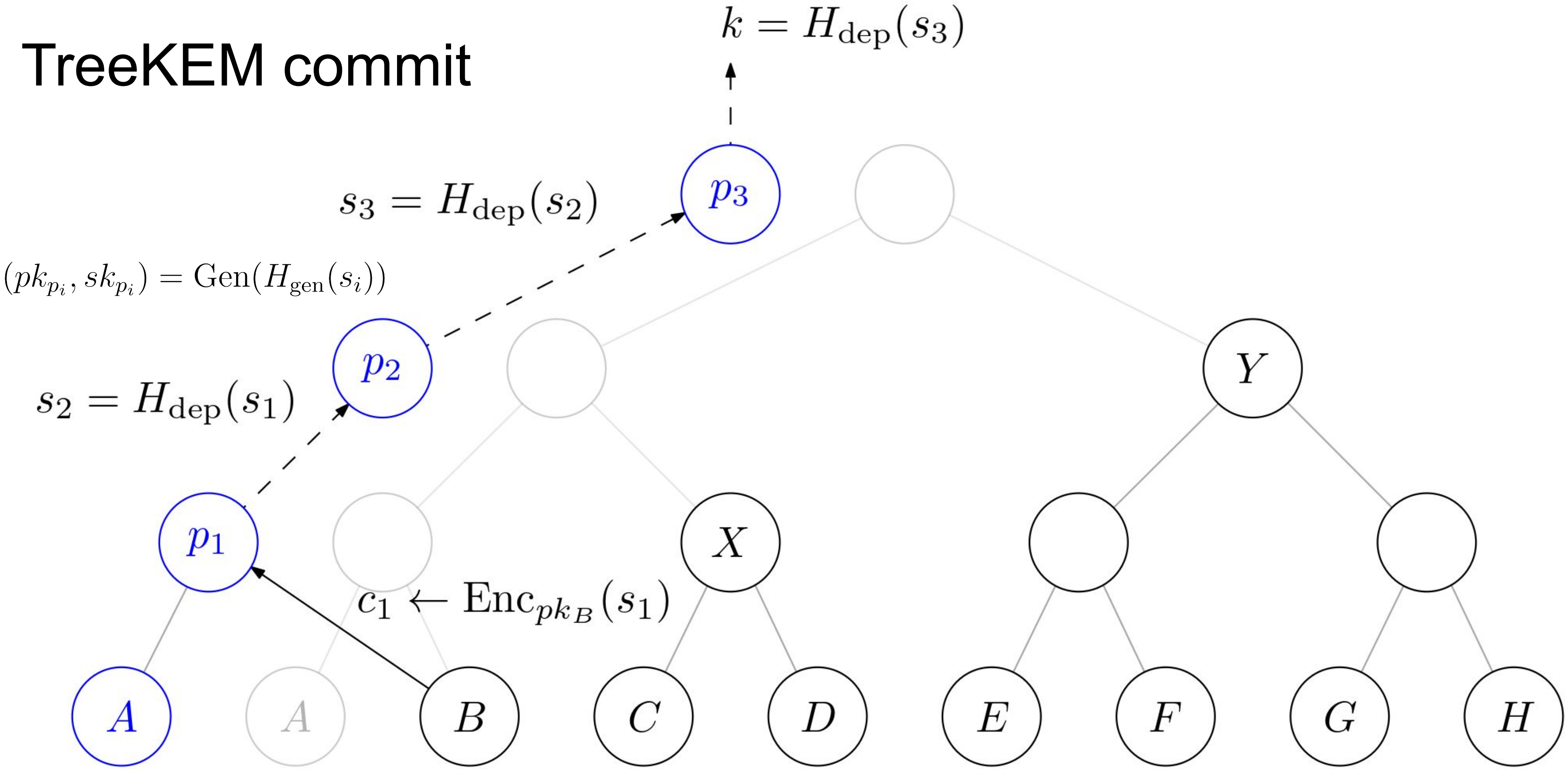
TreeKEM commit



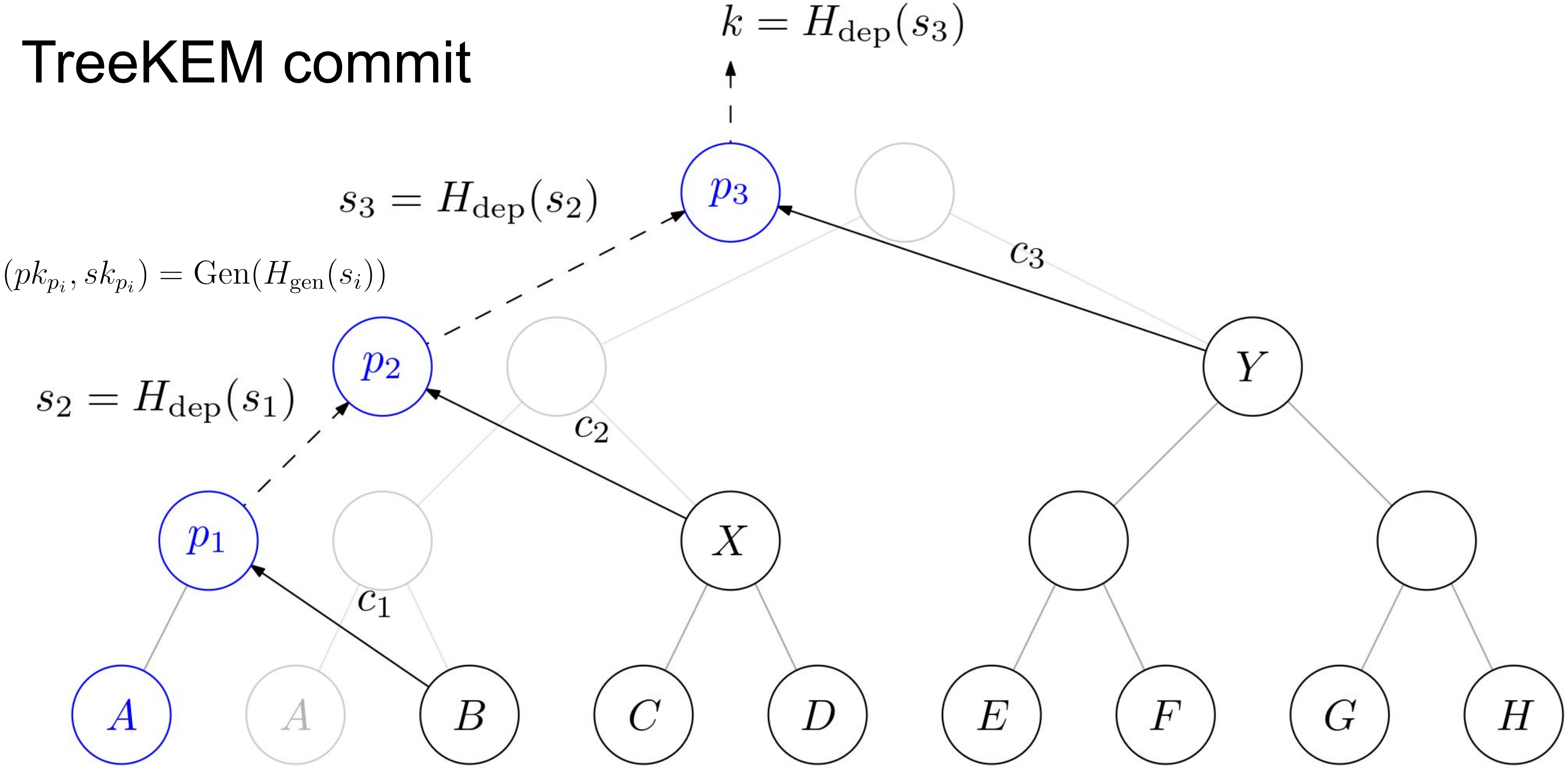
TreeKEM commit



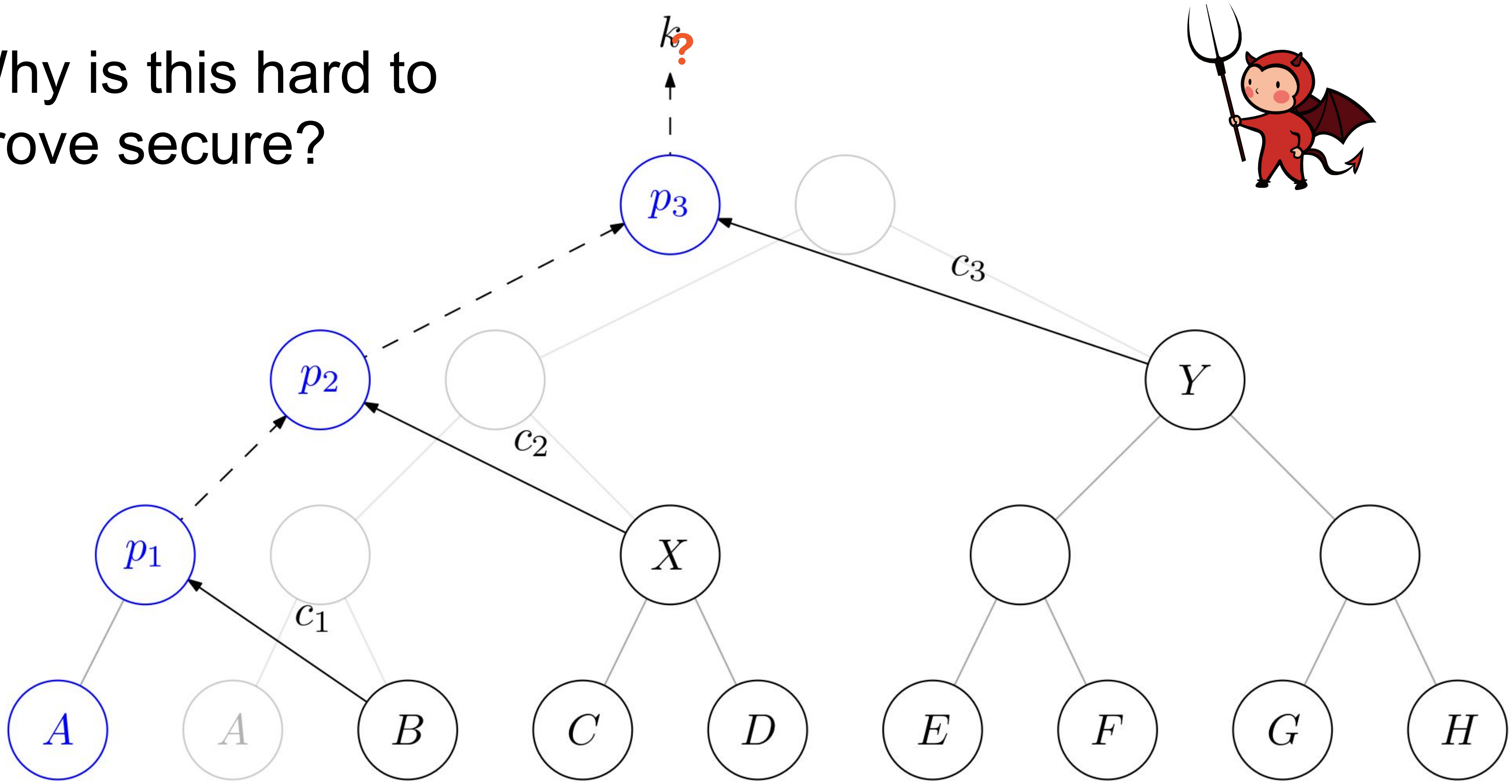
TreeKEM commit



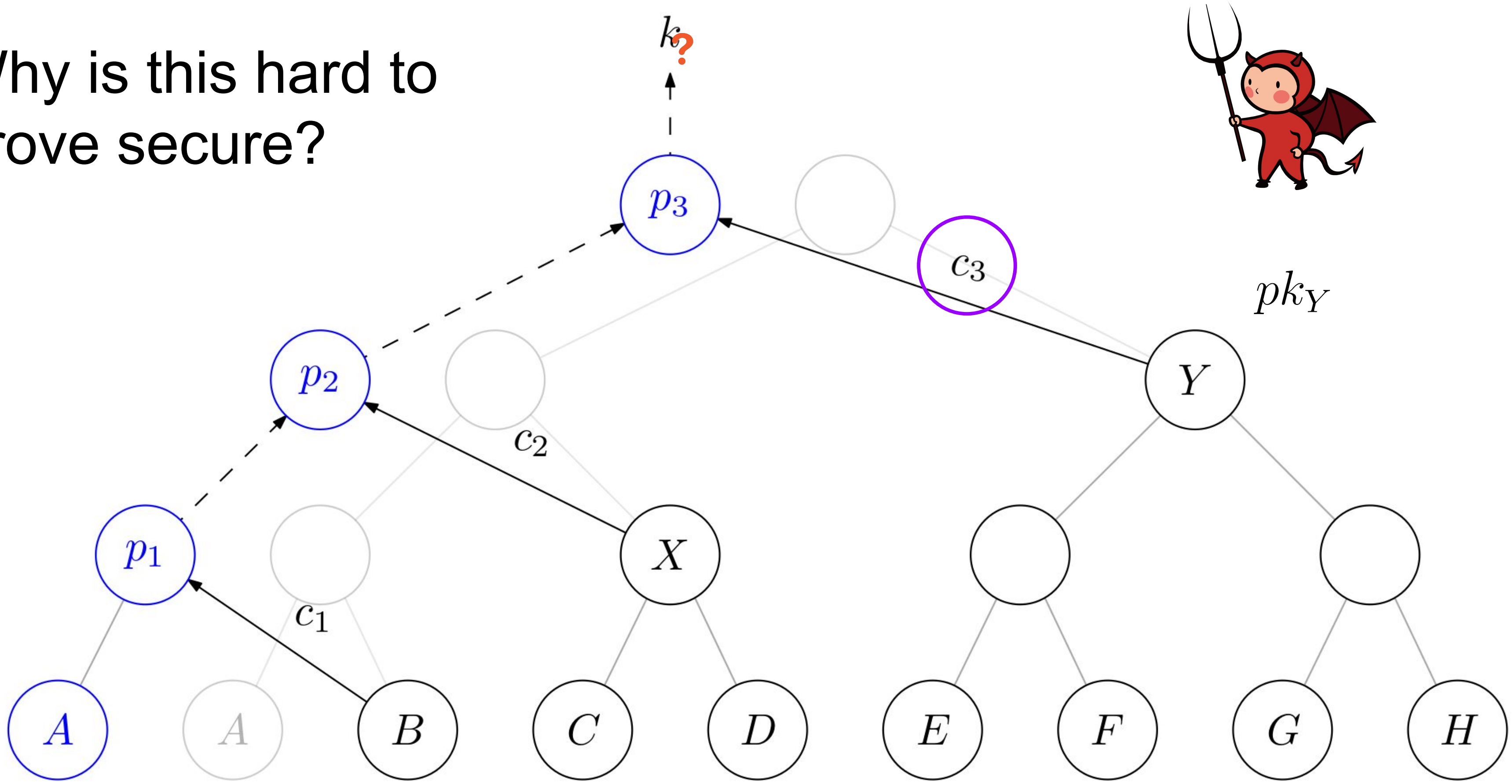
TreeKEM commit



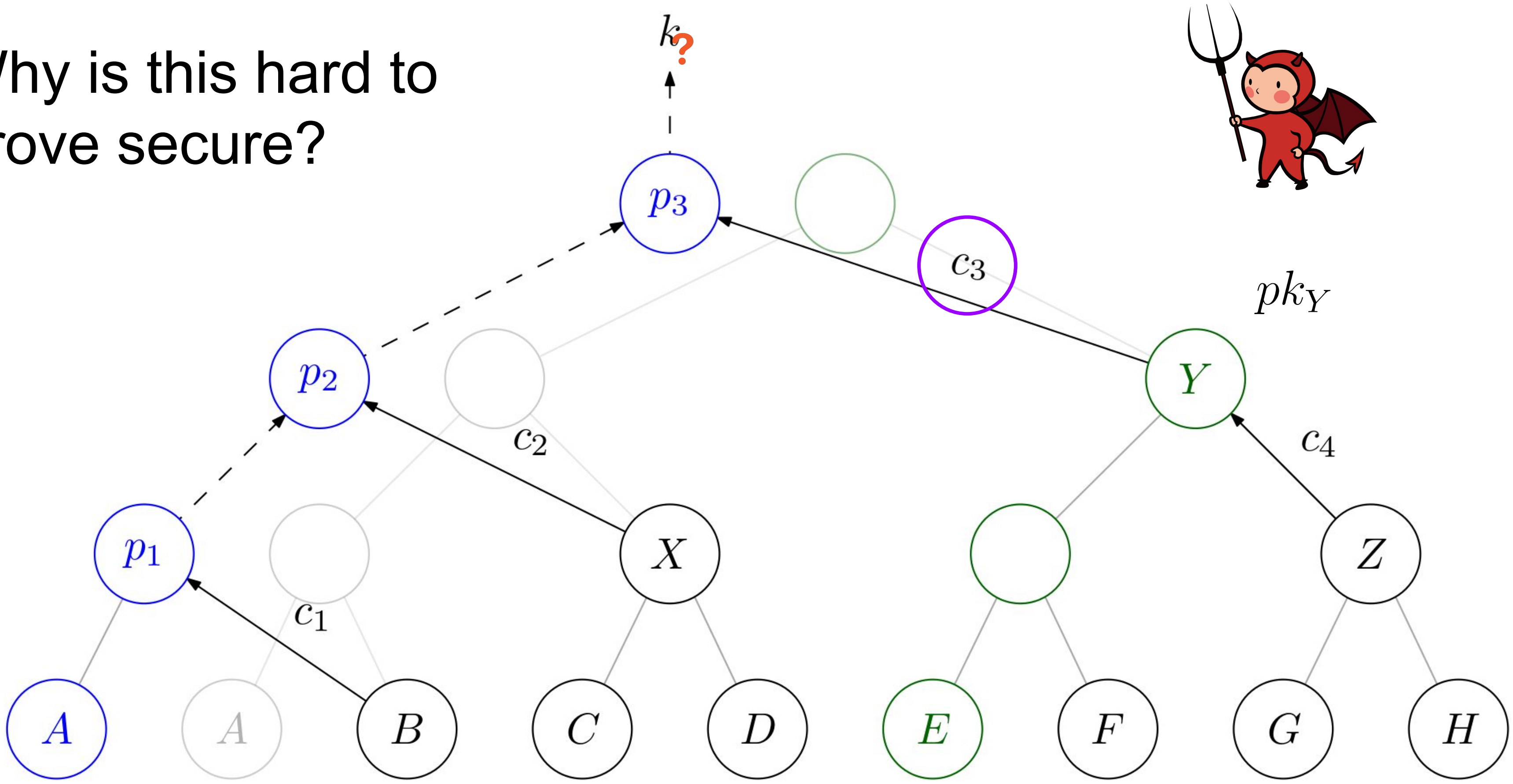
Why is this hard to prove secure?



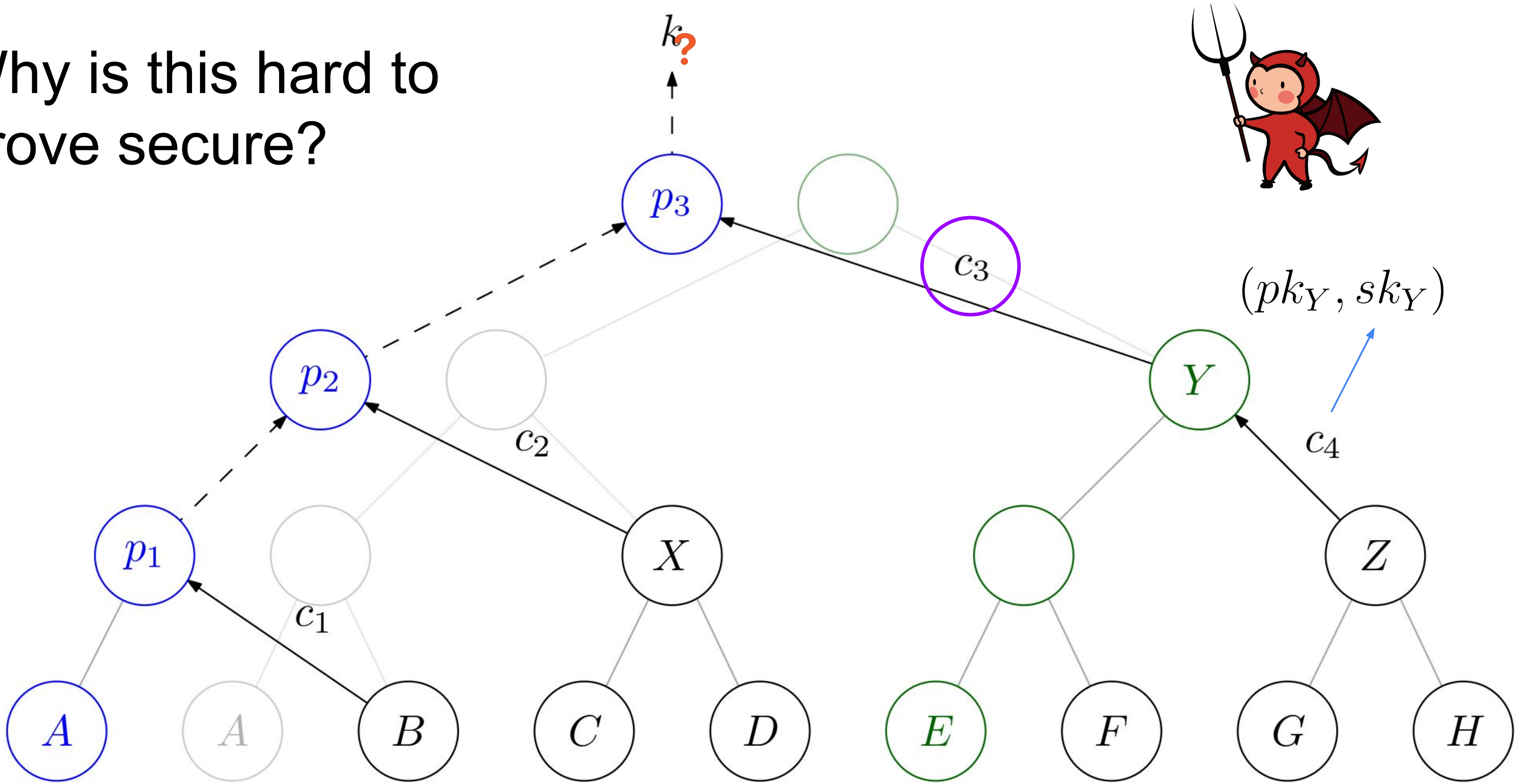
Why is this hard to prove secure?



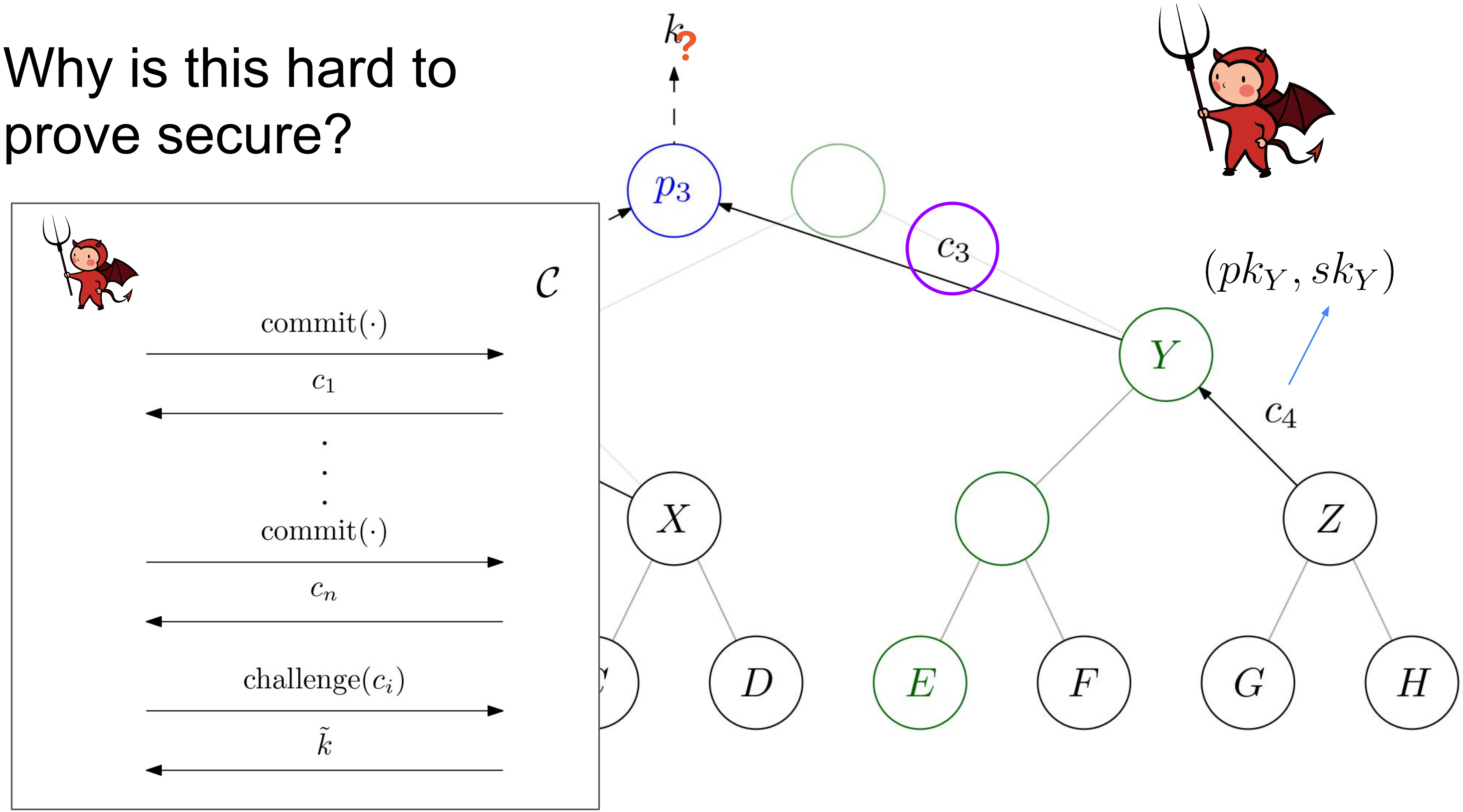
Why is this hard to prove secure?



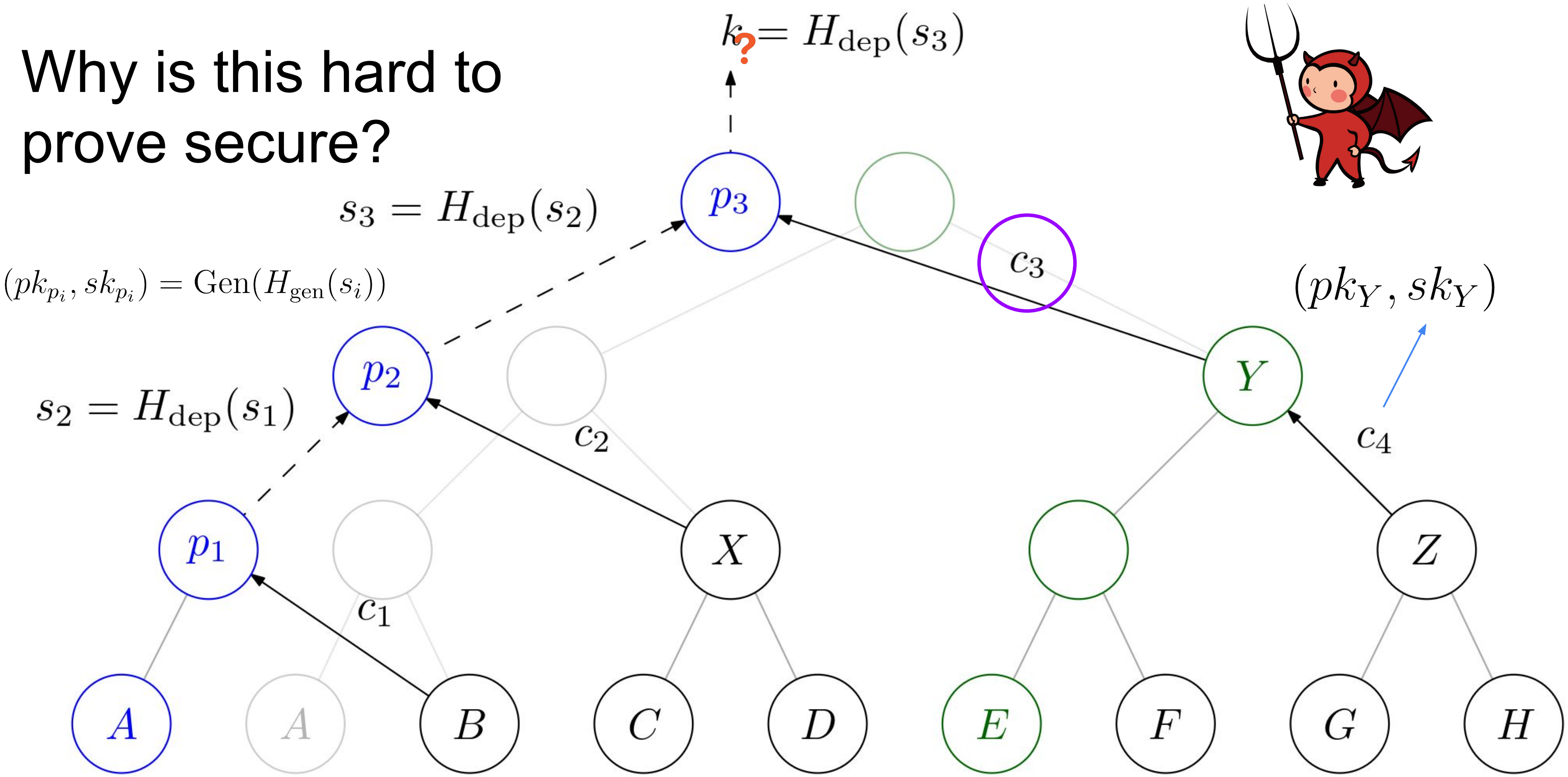
Why is this hard to prove secure?



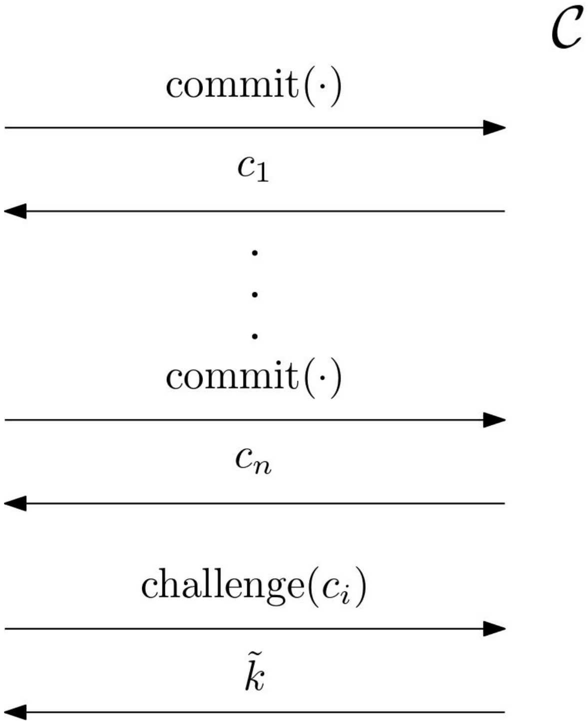
Why is this hard to prove secure?

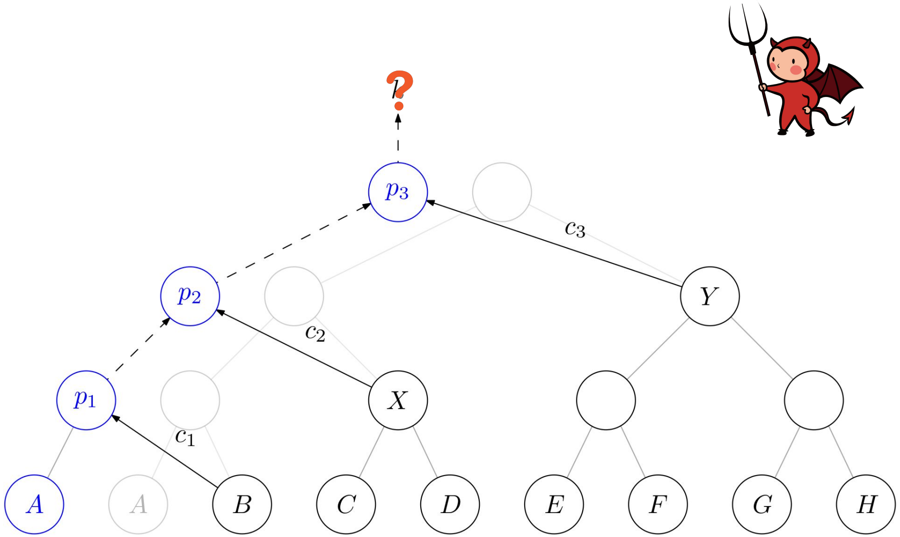


Why is this hard to prove secure?



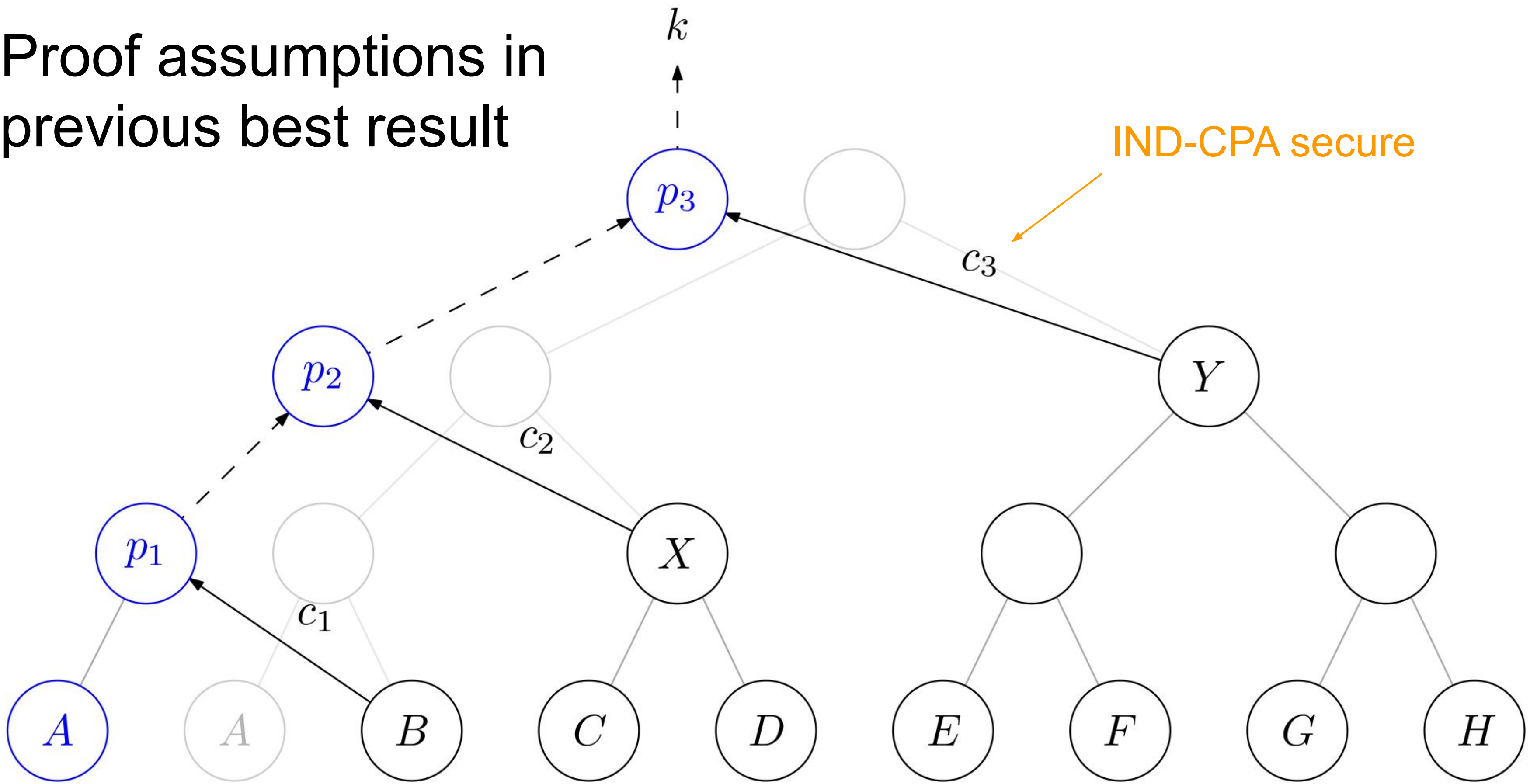
On to the results!



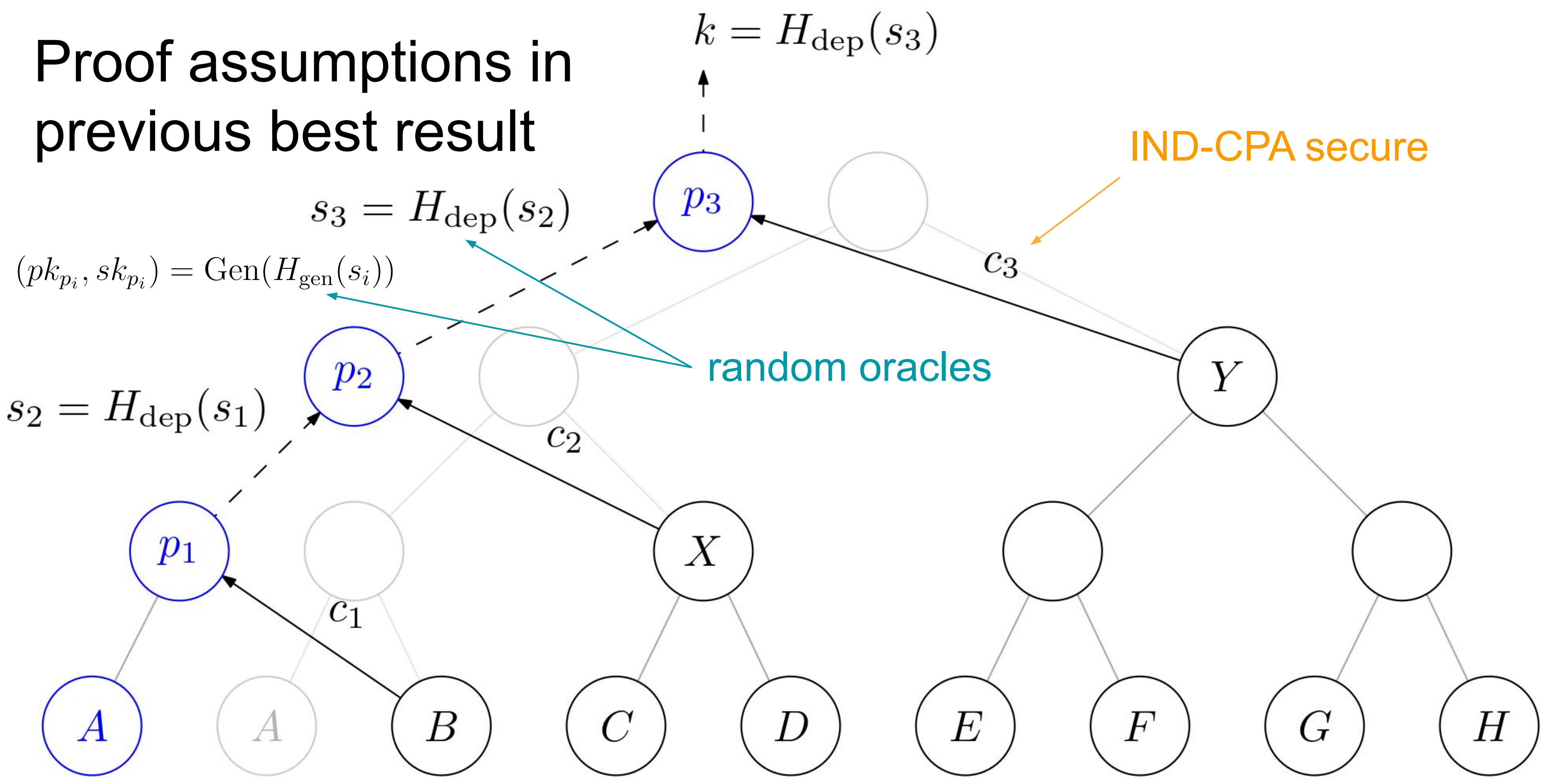


Proof assumptions

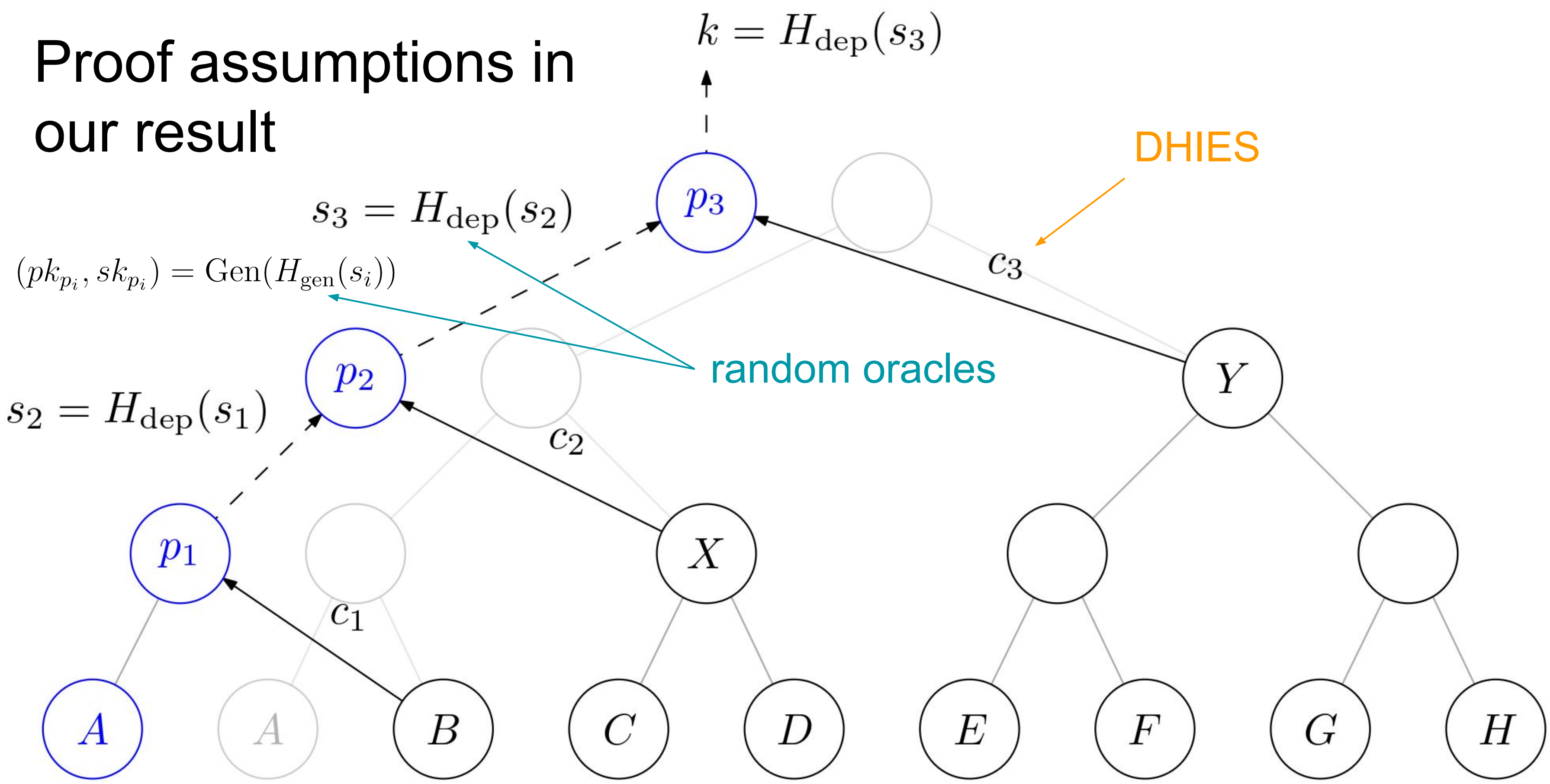
Proof assumptions in previous best result



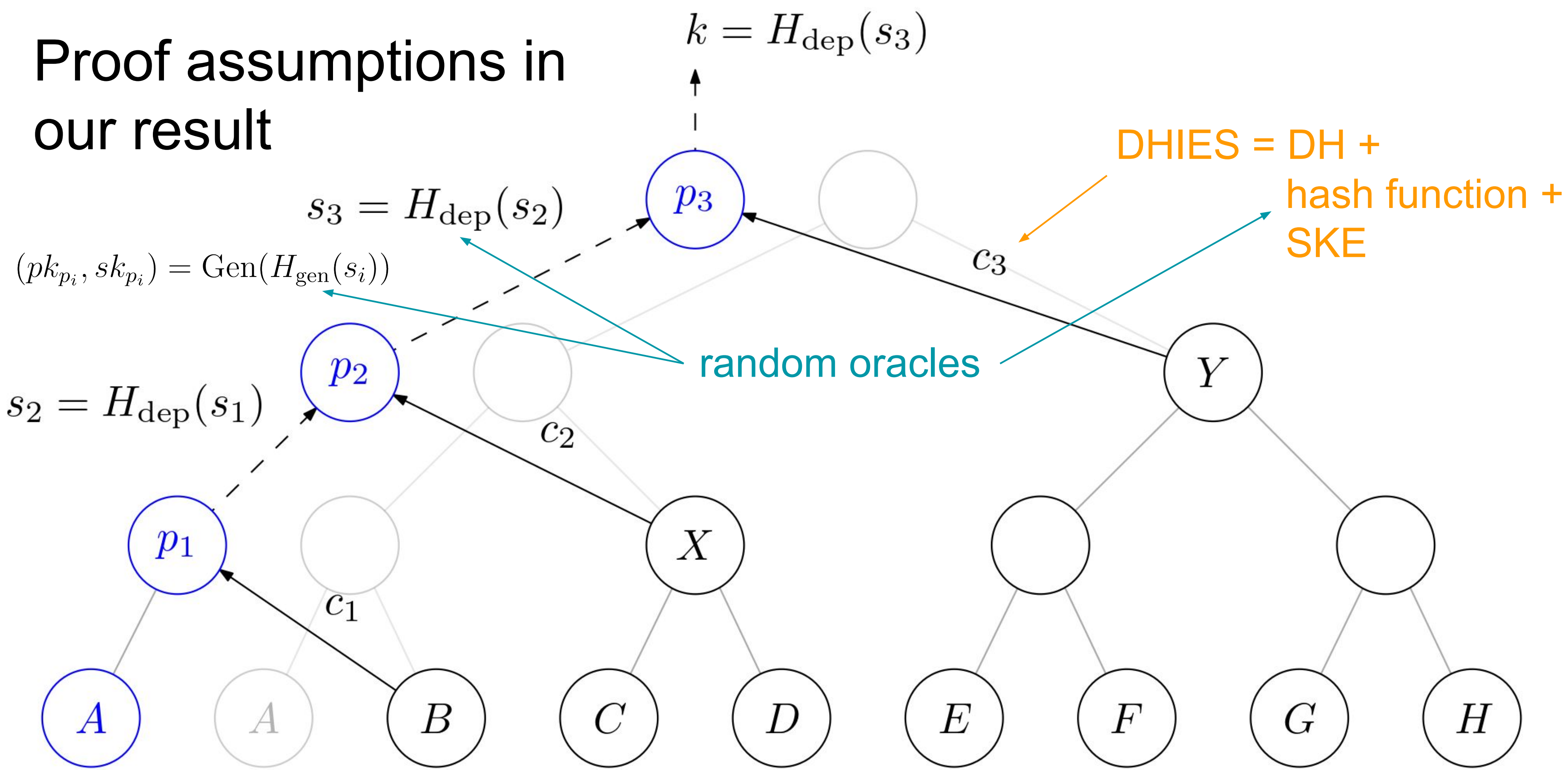
Proof assumptions in previous best result



Proof assumptions in our result

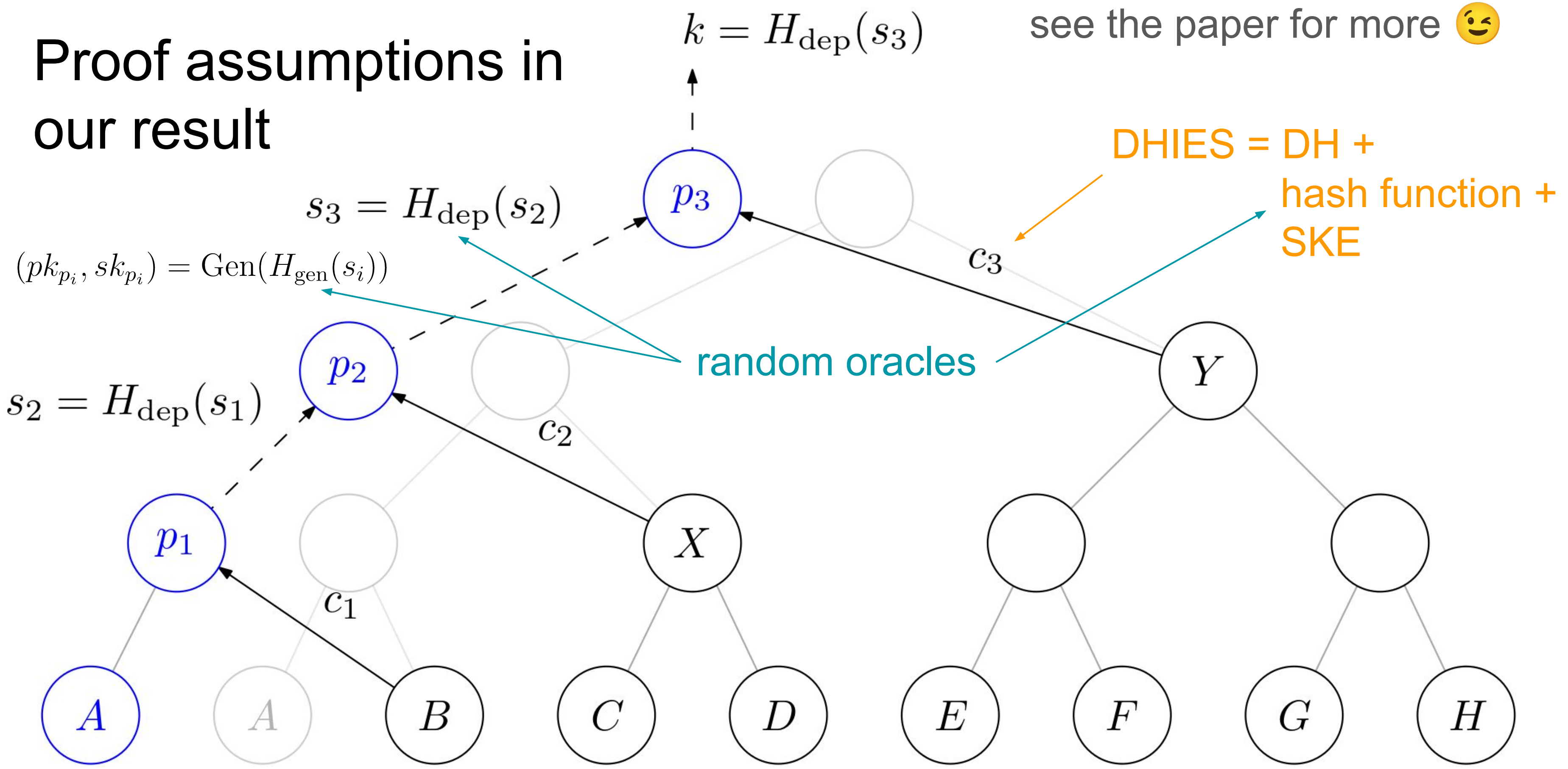


Proof assumptions in our result



Proof assumptions in our result

see the paper for more 😉



The result

c = #commits u = #users

	Few updates	Frequent updates
Updates per commit	$\mathcal{O}(\log(u))$	up to u
Security against compromises	weaker	stronger
Efficiency	better	worse

The result: few updates

$\mathcal{C}$ = #commits $\mathcal{U}$ = #users

The result: few updates

c = #commits u = #users

$$\Pr[\mathcal{A} \text{ breaks TreeKEM}] \leq \mathcal{O}(c^2 \cdot \log(u)^2 \cdot \epsilon_{\text{PKE}}) + \text{negl}$$

The result: few updates

c = #commits u = #users

$c \gg u$!

$$\Pr[\mathcal{A} \text{ breaks TreeKEM}] \leq \mathcal{O}(\underline{c}^2 \cdot \log(u)^2 \cdot \epsilon_{\mathrm{PKE}}) + \mathrm{negl}$$

The result: few updates

c = #commits $\quad$ u = #users

$c \gg u$!

$$\Pr[\mathcal{A} \text{ breaks TreeKEM}] \leq \mathcal{O}(\underline{c}^2 \cdot \log(u)^2 \cdot \epsilon_{\text{PKE}}) + \text{negl}$$

vs.

$$\Pr[\mathcal{A} \text{ breaks TreeKEM}] \leq \mathcal{O}(u \cdot \underline{c} \cdot \log u \cdot \epsilon_{\text{SKE}} + \underline{c} \cdot \log u \cdot \epsilon_{\text{DH}}) + \text{negl}$$

The result: frequent updates

c = #commits u = #users

$c \gg u$!

$$\Pr[\mathcal{A} \text{ breaks TreeKEM}] \leq \mathcal{O}(\underline{c}^2 \cdot u^2 \cdot \epsilon_{\mathrm{PKE}}) + \mathrm{negl}$$

vs.

$$\Pr[\mathcal{A} \text{ breaks TreeKEM}] \leq \mathcal{O}(\underline{c} \cdot u^2 \cdot \epsilon_{\mathrm{SKE}} + \underline{c} \cdot u \cdot \epsilon_{\mathrm{DH}}) + \mathrm{negl}$$

Interpreting the result

Consider a group chat with **10’000 users**, making **one commit/hour** for **5 years** with **128-bit parameters**

Interpreting the result

Consider a group chat with **10’000 users**, making **one commit/hour** for **5 years** with **128-bit parameters**

	Few updates	Frequent updates
Previous result	82 bits	64 bits
Our result	90 bits	81 bits

Interpreting the result

DHIES = DH +
hash function +
SKE

Consider a group chat with **10’000 users**, making **one commit/hour** for **5 years** with **128-bit parameters** but **256-bit SKE**

Interpreting the result

DHIES = DH + hash function + SKE

Consider a group chat with **10’000 users**, making **one commit/hour** for **5 years** with **128-bit parameters** but **256-bit SKE**

	Few updates	Frequent updates
Previous result	82 bits	64 bits
Our result	~~90~~ 104 bits	~~81~~ 95 bits

Main takeaway

We've proven security for TreeKEM with practical parameters 🥳

Main takeaway

We’ve proven security for TreeKEM with practical parameters 🥳

… but not yet for MLS as a whole 🙁