Kintsugi 🔑🏺

Decentralized E2EE Key Recovery

Emilie Ma & Martin Kleppmann

Slides available at emilie.ma/fosdem2025 ◦ hello@emilie.ma

I lost my phone. What now?

- with non-E2EE apps: log in with the same username/password
- with E2EE apps: server doesn’t store a copy of key
    - recovery PIN
    - recovery contact
    - recovery codes
    - recovery files
    - and more…

Existing schemes have tradeoffs.

Recovery PINs	**Recovery Contacts**	**Recovery Codes/ Files**
• e.g. Signal SVR, WhatsApp • Requires secure hardware for rate-limiting guesses (otherwise, brute-forceable)	• e.g. Apple iCloud, PreVeil • Have to totally trust contacts • Usually can collude to gain access to your account	• e.g. LastPass, Bitcoin • Protects against brute-force/guessing because high-entropy, but requires keeping a copy

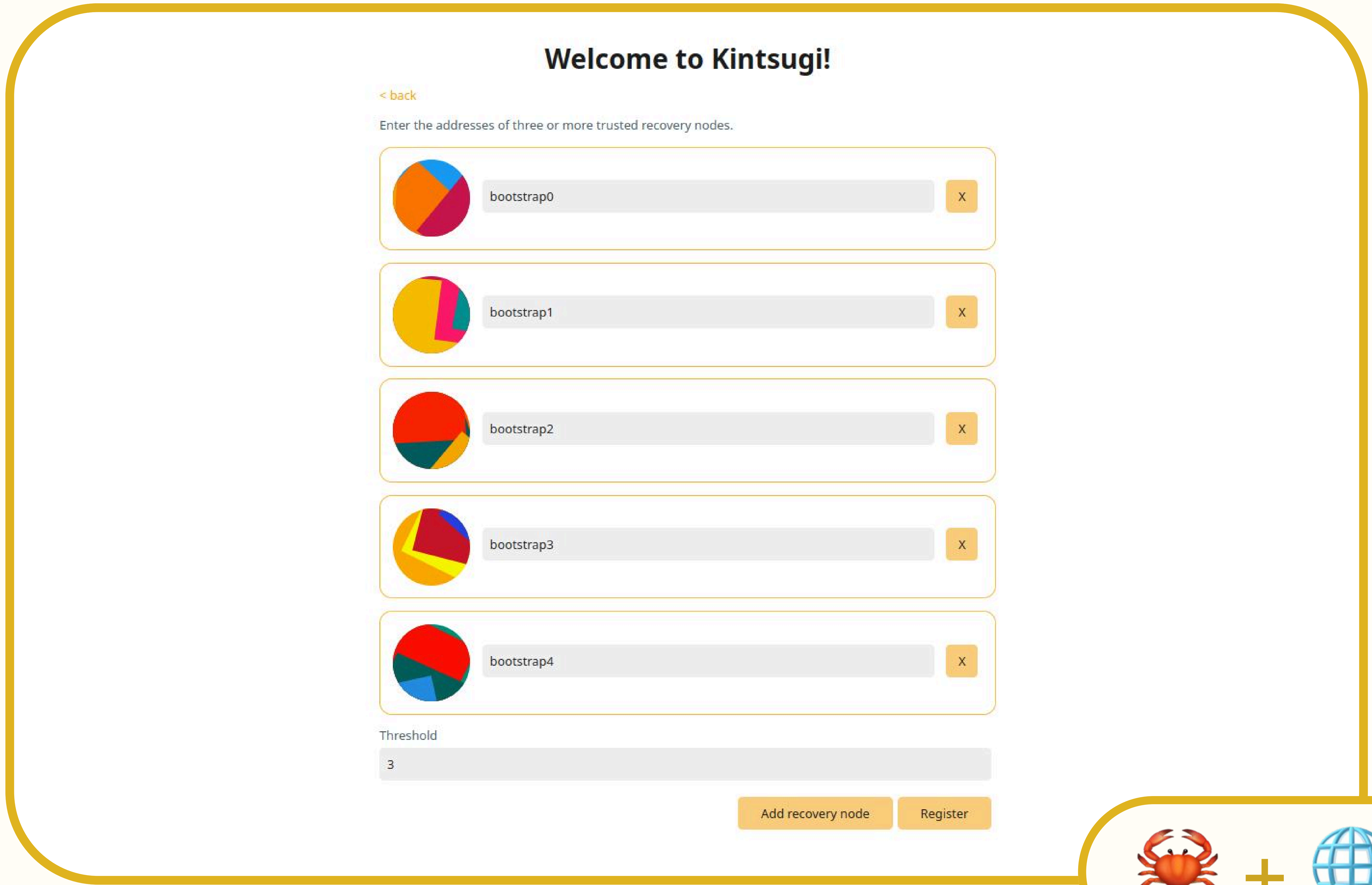
Centralization doesn’t always work.

- some applications require metadata privacy (e.g. Tor)
- others may have infrastructure shut down (e.g. sanctioned activists)
- services may lack/want to avoid central authority group
- infrastructure can be cost-prohibitive
- other issues: single point of trust, infra availability

Introducing Kintsugi!

- decentralized key recovery protocol based on P2P network
  - recovery servers + contacts’ devices + a mix
- recovery by contacting some threshold *t+1* of recovery nodes
  - each hold share of secret for user to recover key
- users can update recovery nodes at any time
- protects against brute-forcing low-entropy password
- also protects against colluding, “honest-but-curious” recovery nodes

Demo



Welcome to Kintsugi!

< back

Enter the addresses of three or more trusted recovery nodes.

bootstrap0

bootstrap1

bootstrap2

bootstrap3

bootstrap4

Threshold

3

🦀 + 🌐 + ⚛️

github.com/kewbish/kintsugi

What’s an OPRF?

- Oblivious Pseudo-Random Function
    - user keeps a secret value, *U*
    - server keeps a secret value, *S*
    - user learns the result F(U, S) (but **not S**), server learns nothing

👤 🎩

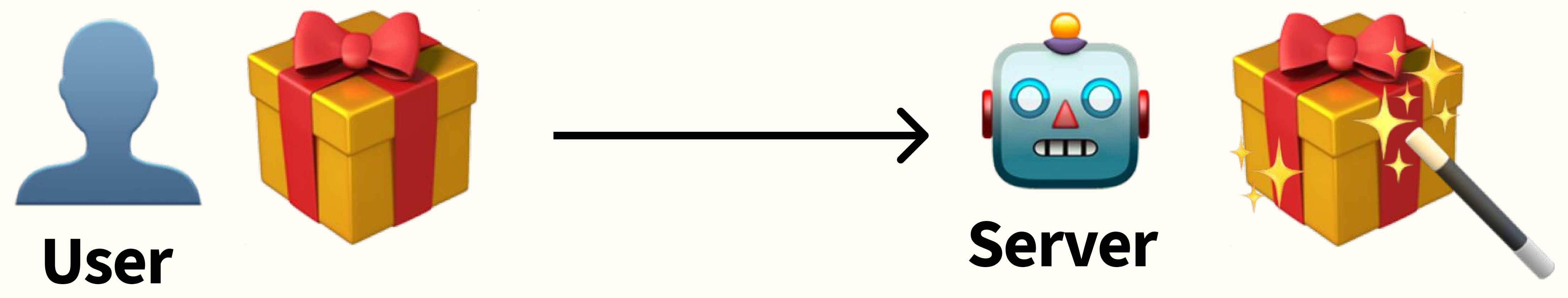
User

What's an OPRF?

- Oblivious Pseudo-Random Function
  - user keeps a secret value, *U*
  - server keeps a secret value, *S*
  - user learns the result F(U, S) (but **not S**), server learns nothing

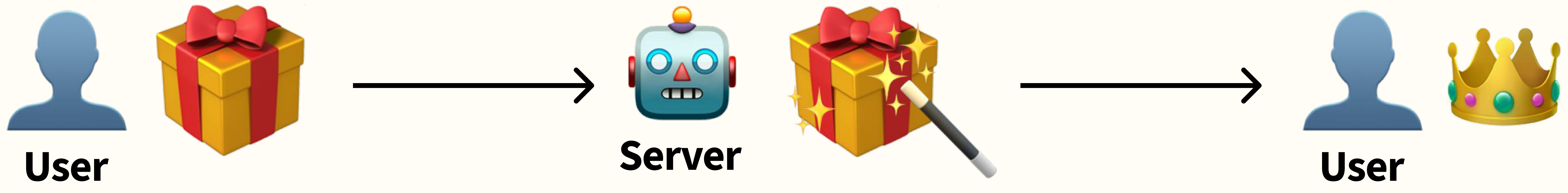
What's an OPRF?

- Oblivious Pseudo-Random Function
  - user keeps a secret value, *U*
  - server keeps a secret value, *S*
  - user learns the result F(U, S) (but **not S**), server learns nothing



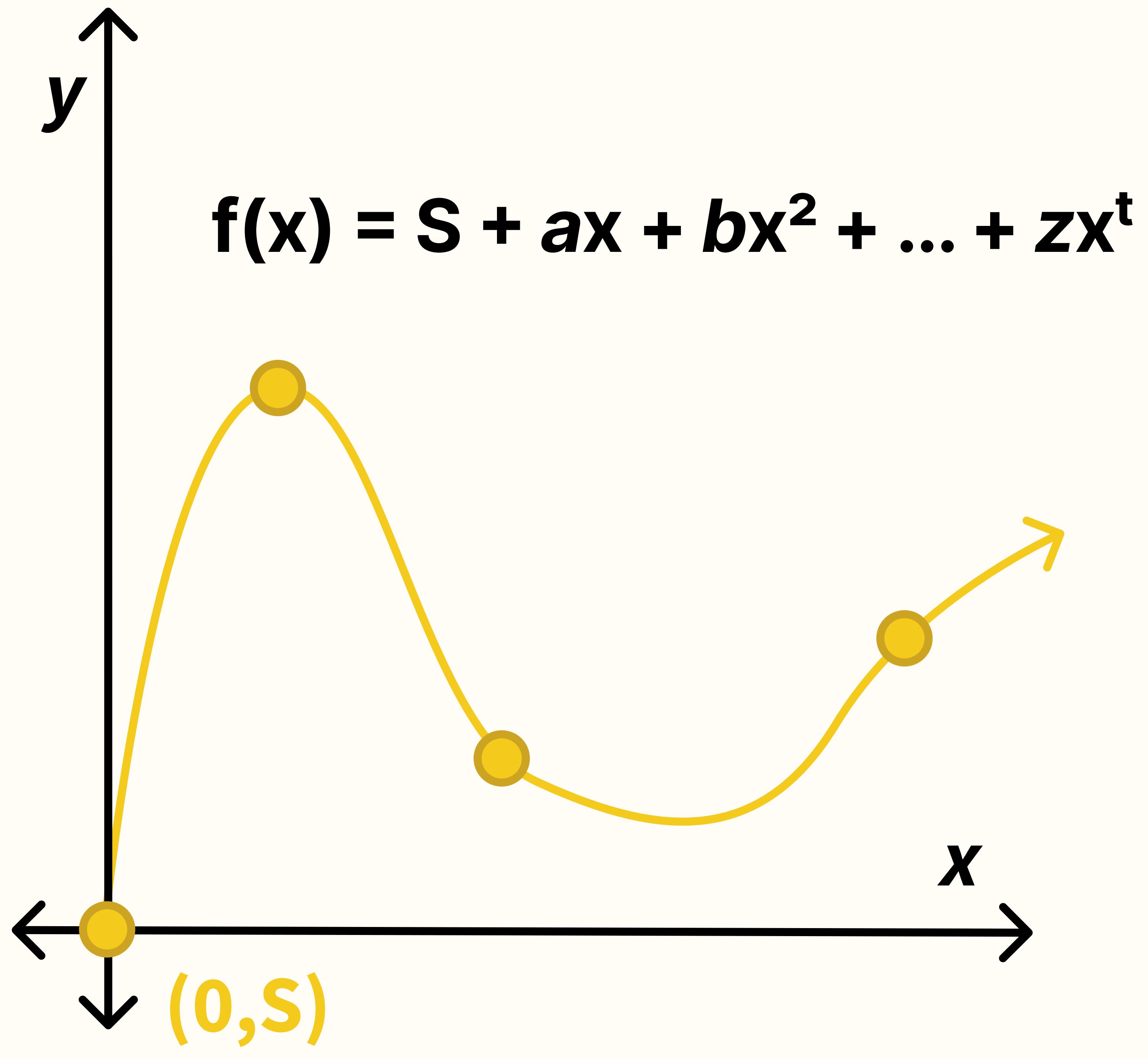
What's an OPRF?

- Oblivious Pseudo-Random Function
    - user keeps a secret value, *U*
    - server keeps a secret value, *S*
    - user learns the result F(U, S) (but **not S**), server learns nothing



Shamir Secret Sharing

- have a secret *S* that you want to split up into *shares*
- require at least *t+1* shares to reconstruct *S*



- each of these points is a share
- can "connect the dots" with enough shares to find the unique function (Lagrange interpolation)
- then can compute f(0) = S

Protocols Used

- combination of:
    - threshold OPRFs (TOPPSS by Jarecki et al.)
        - imagine an OPRF but with multiple “servers”, where you need to reach at least *t+1*
    - dynamic, proactive secret sharing (Honey Badger by Das et al.)
        - recovery nodes can be changed on demand
        - imagine SSS but you can exchange nodes’ shares while keeping *s* the same

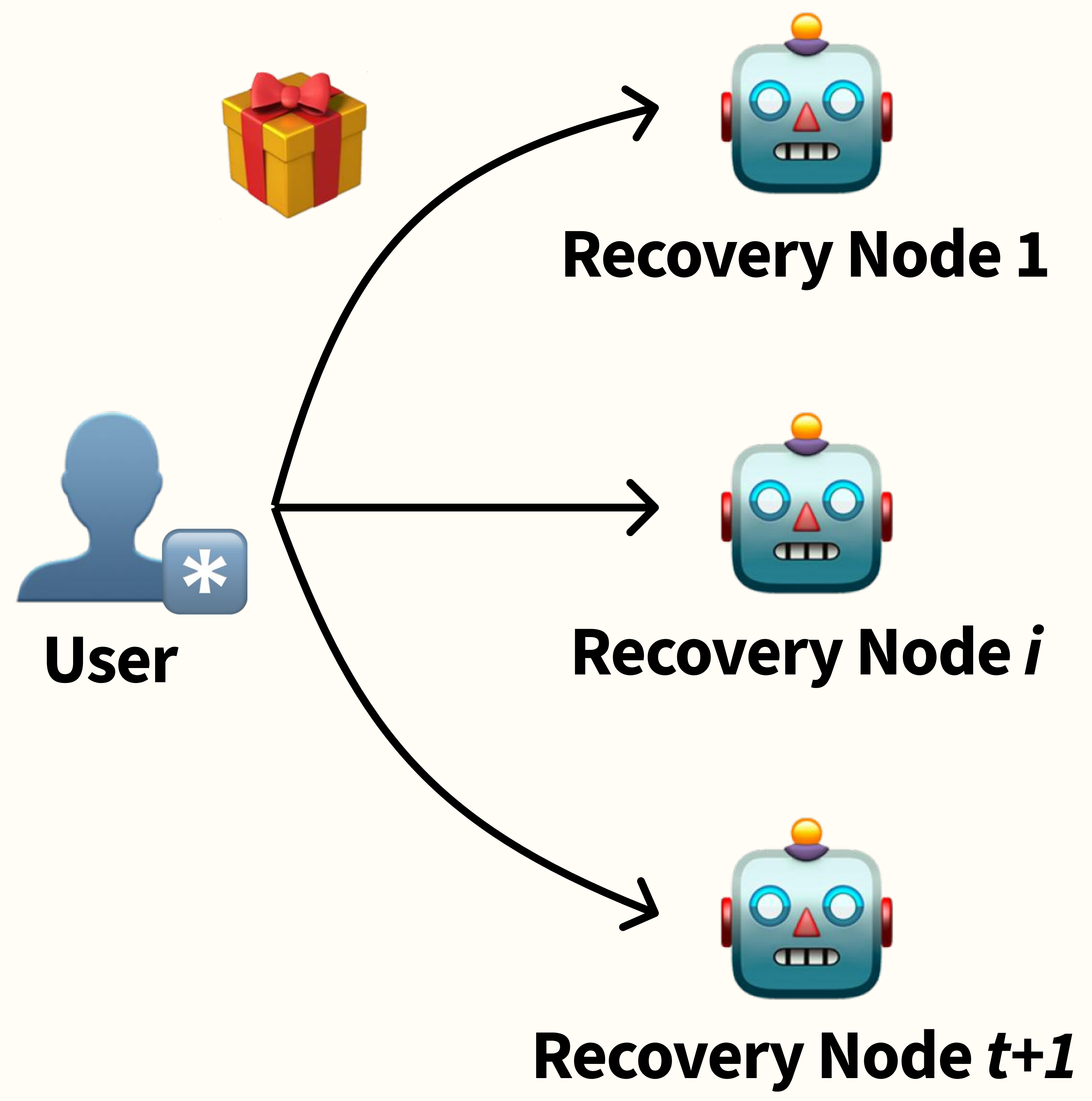
Registration Flow

👤 ✳️

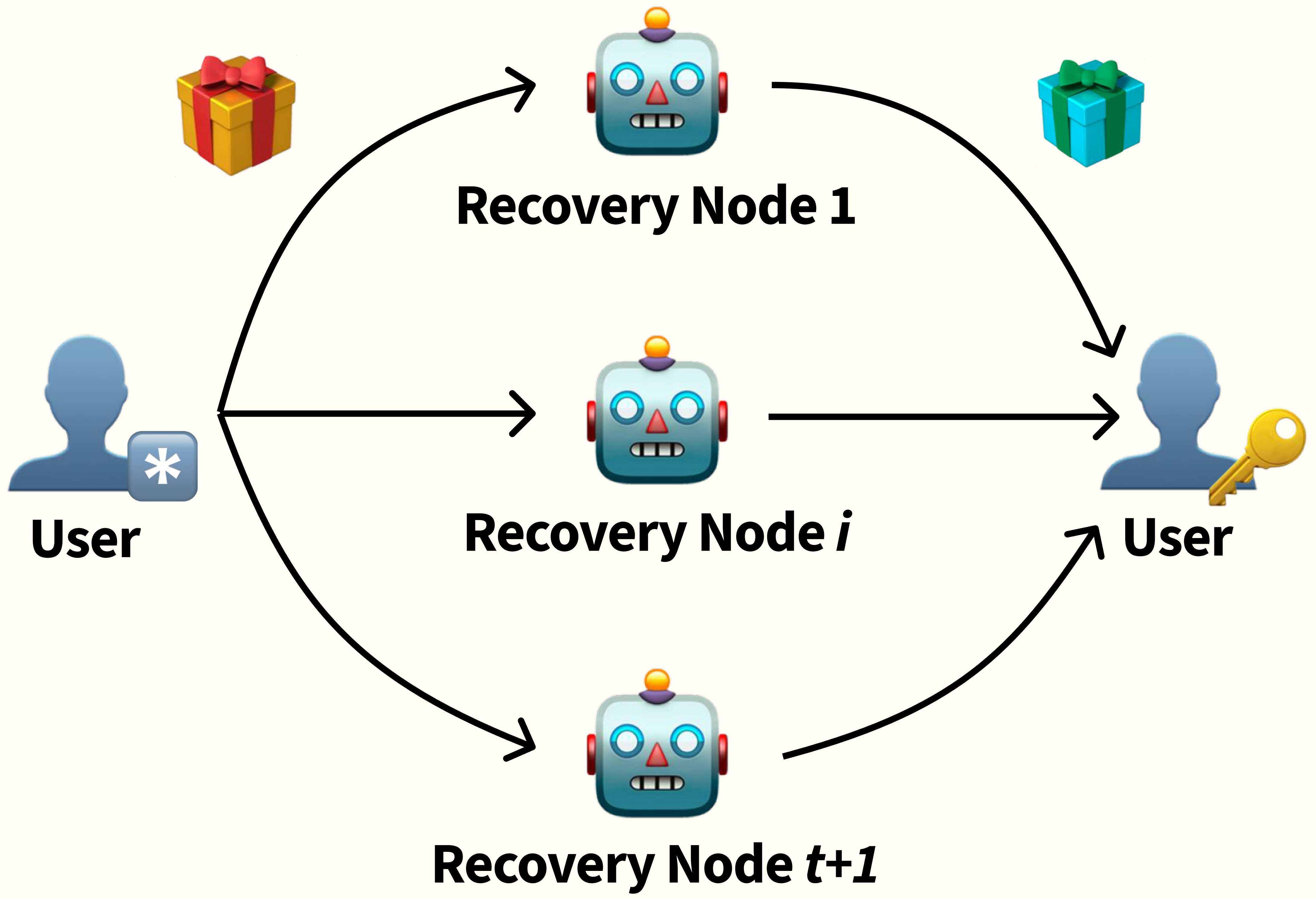
User

Registration Flow

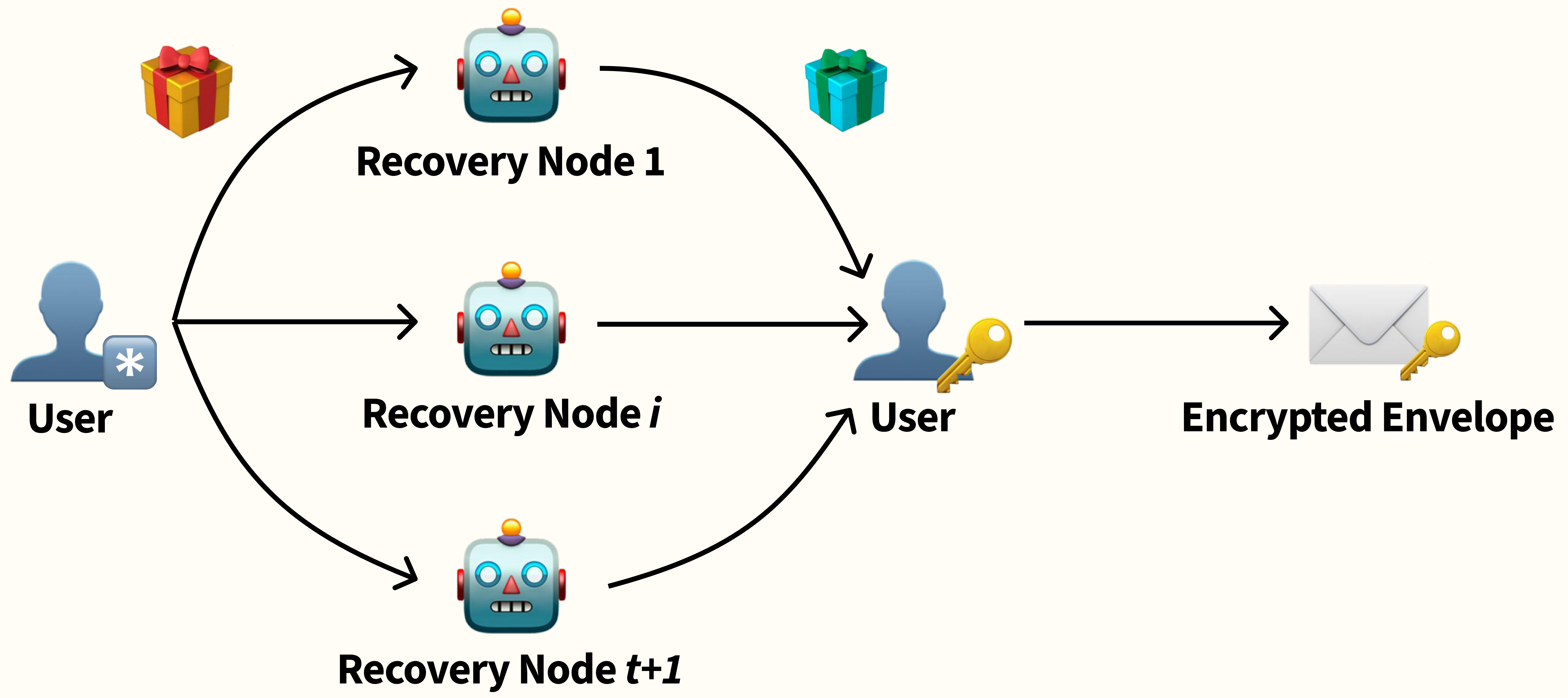
Registration Flow



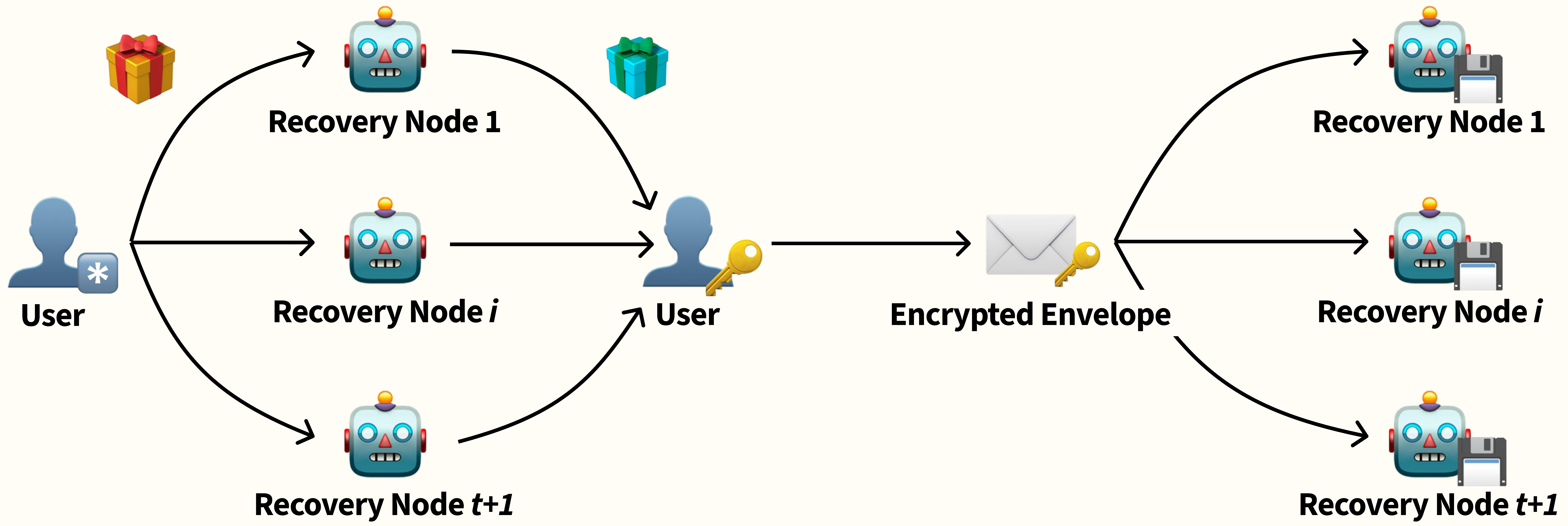
Registration Flow



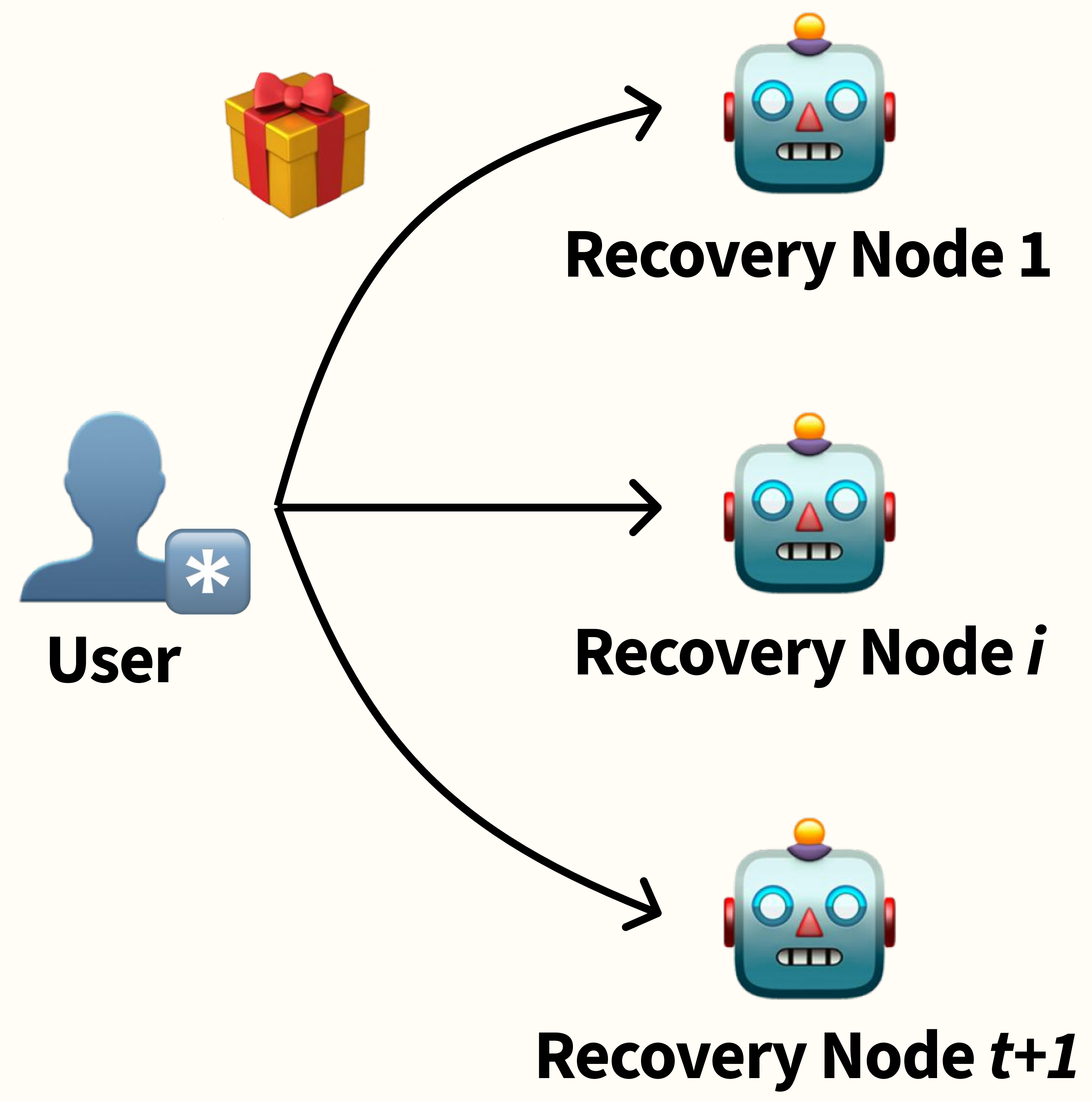
Registration Flow



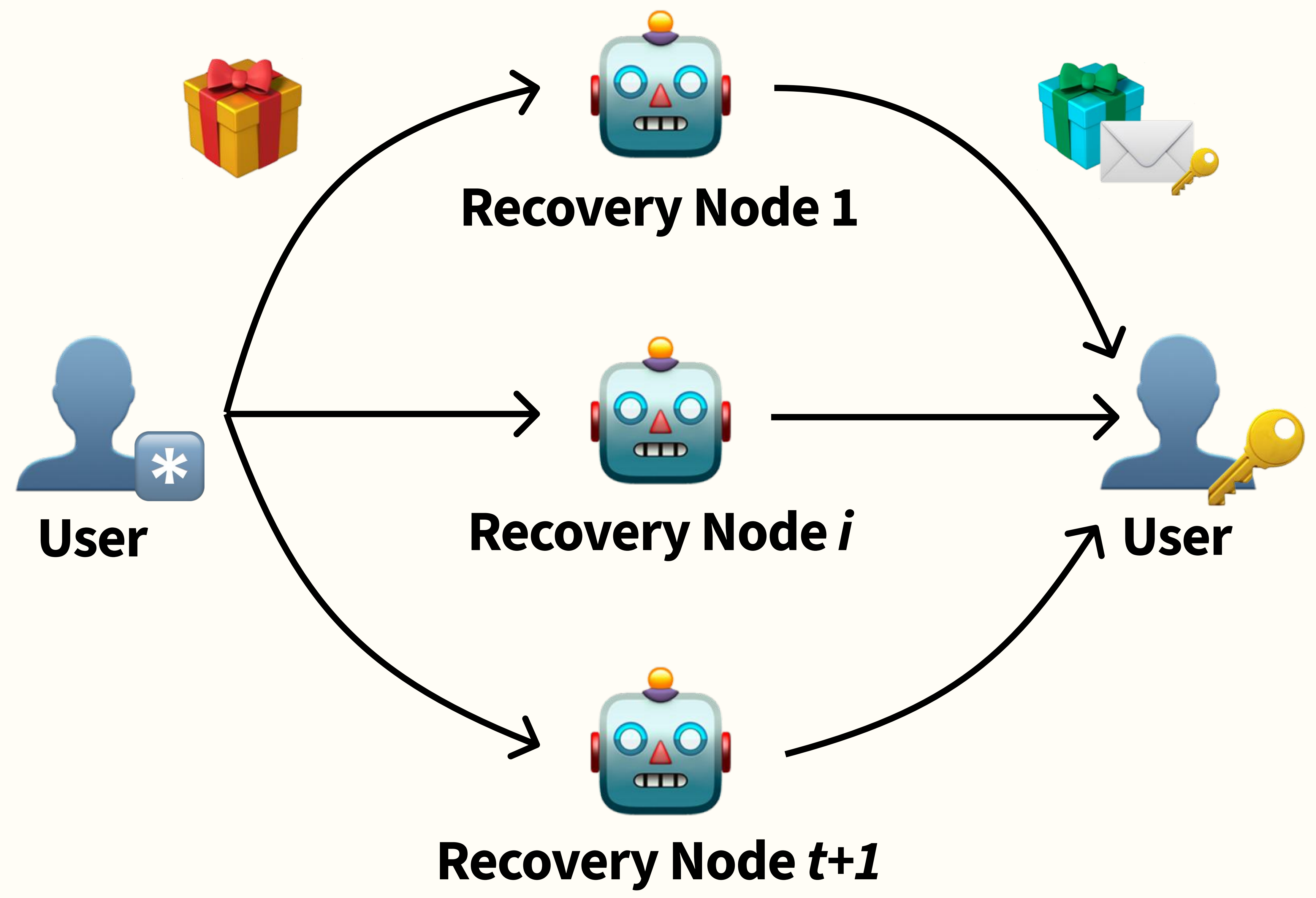
Registration Flow



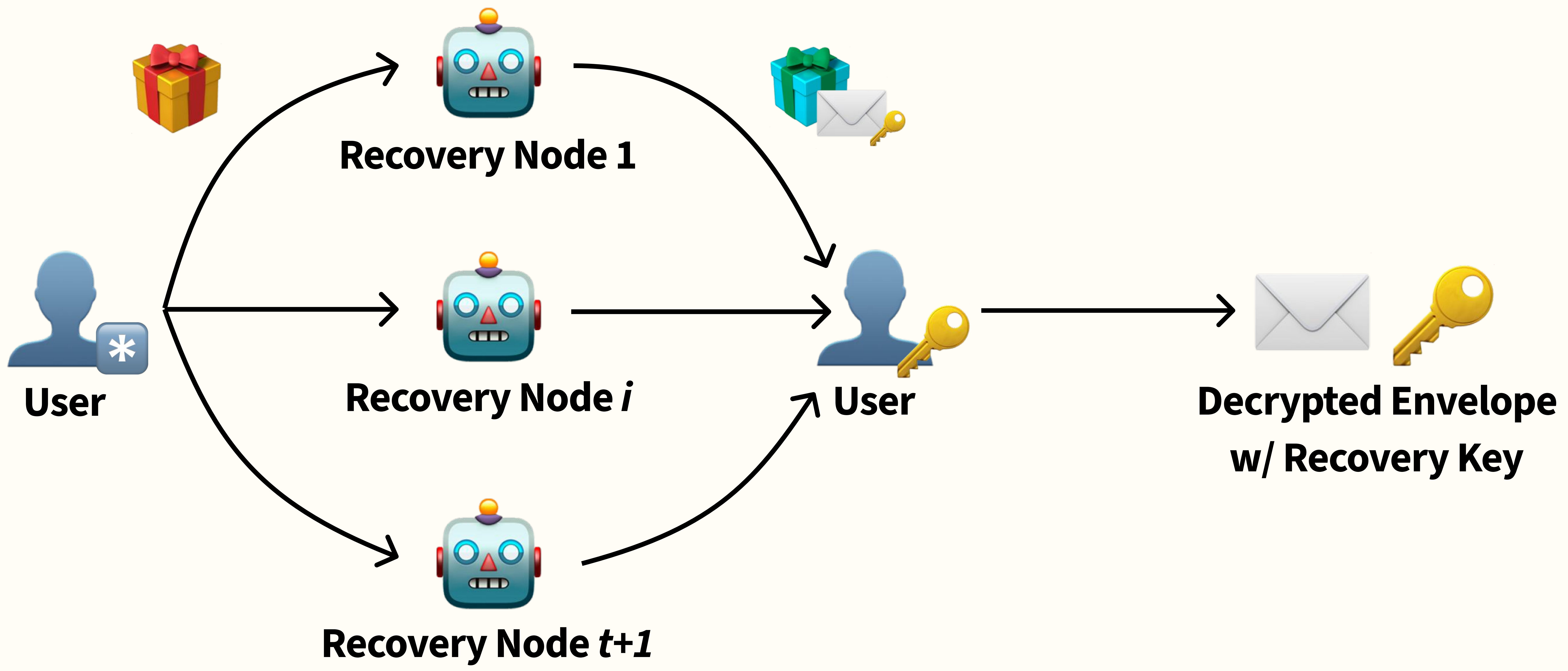
Recovery Flow



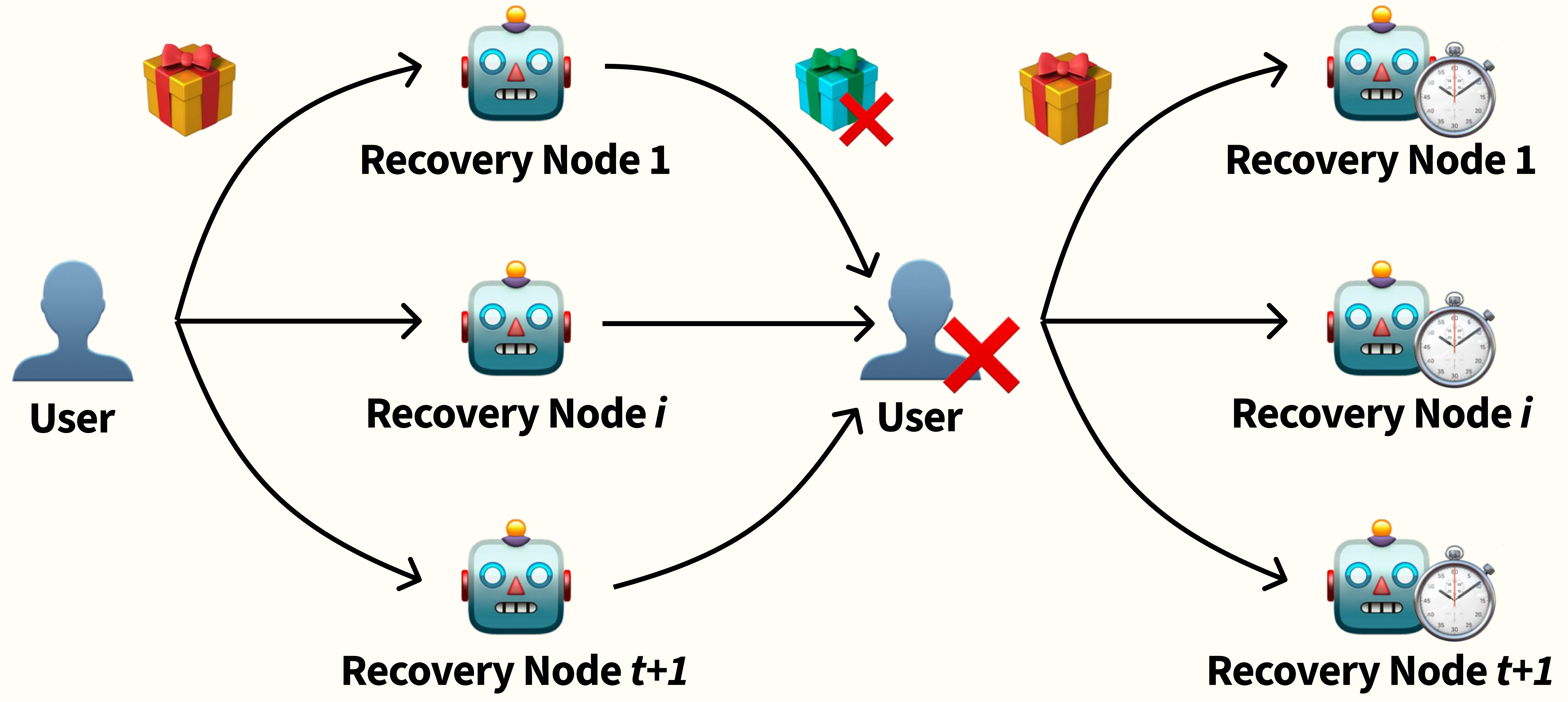
Recovery Flow



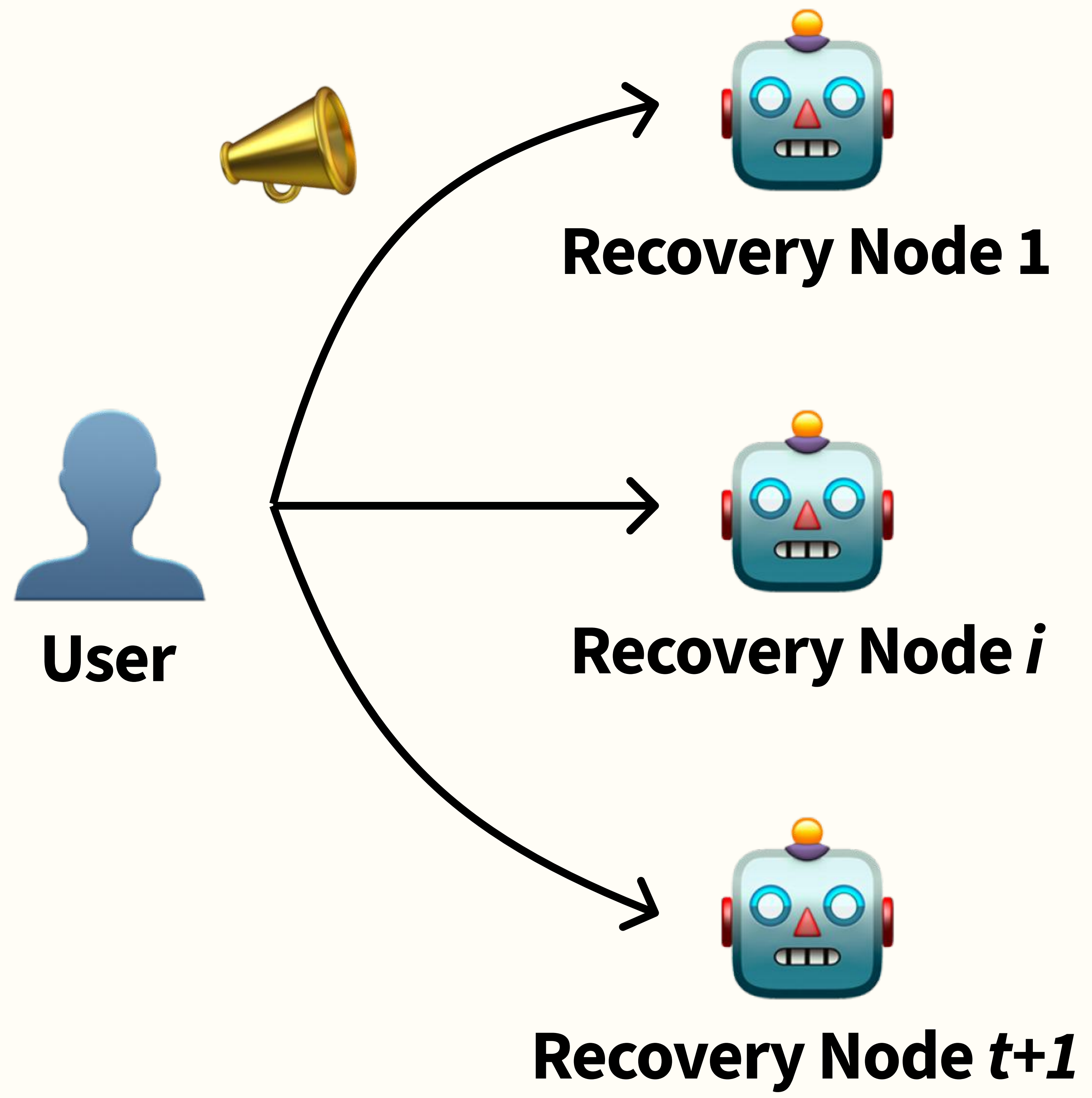
Recovery Flow



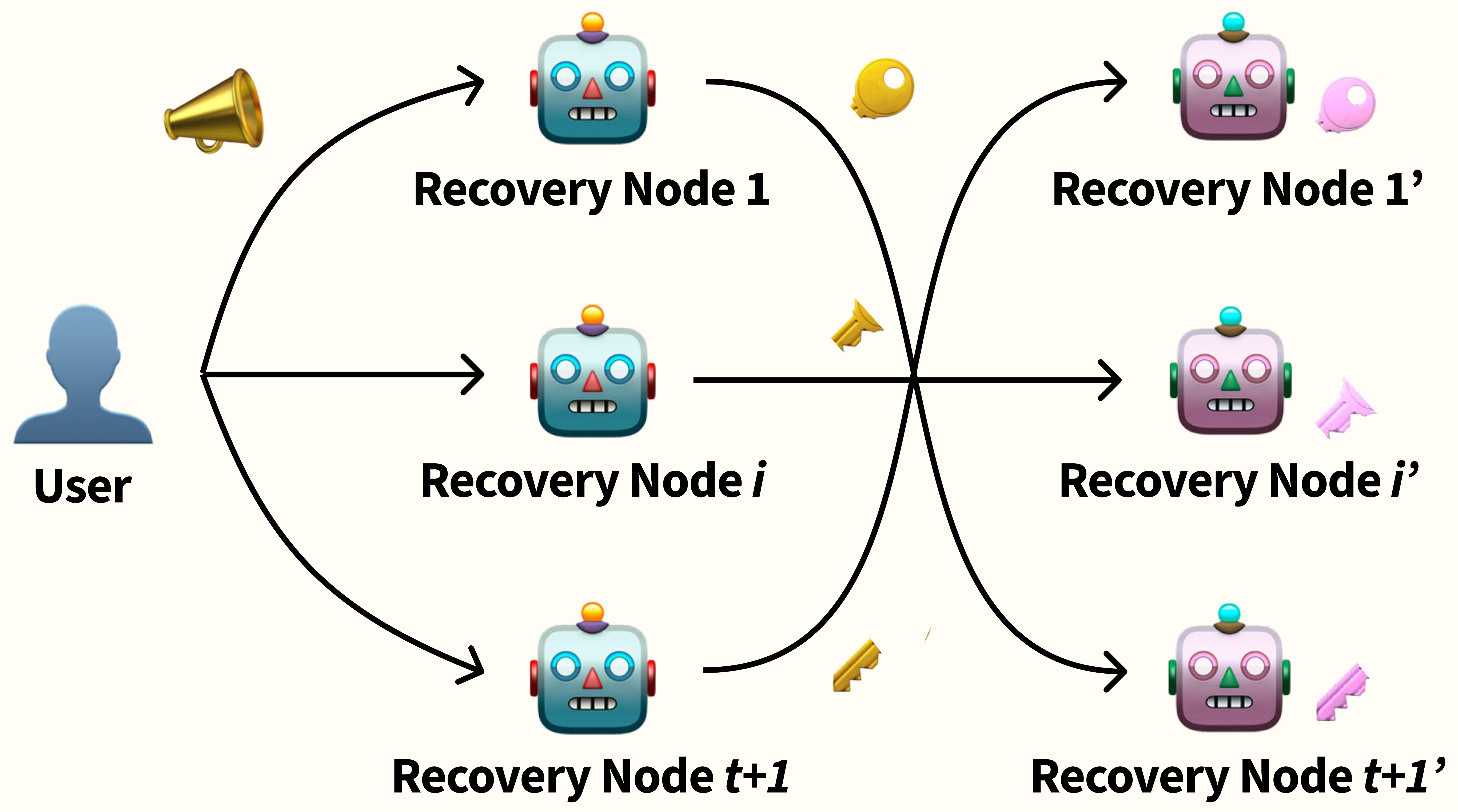
Recovery Flow



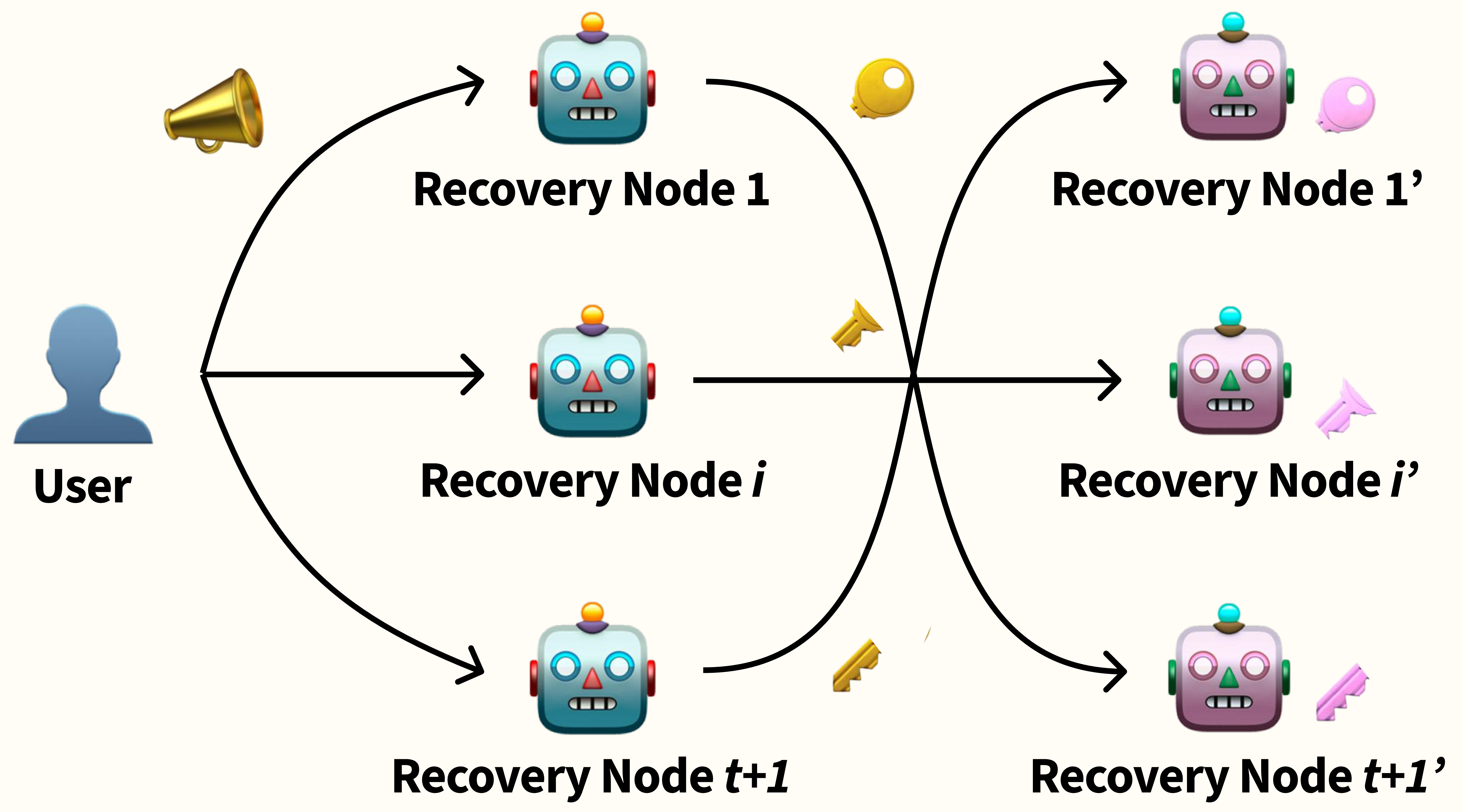
Recovery Node Update Flow



Recovery Node Update Flow



Recovery Node Update Flow



https://emilie.ma/blog/posts/241229/

TL;DR: Kintsugi provides decentralized secure recovery.

- improvements on existing methods:
  - decentralized!
  - no expensive hardware required
  - works in the case of device loss
  - protects against brute-force + colluding recovery nodes
- currently: initial implementation finished
- next: integrating w/ Ink & Switch Beehive project, polishing

Slides available at emilie.ma/fosdem2025 ◦ hello@emilie.ma