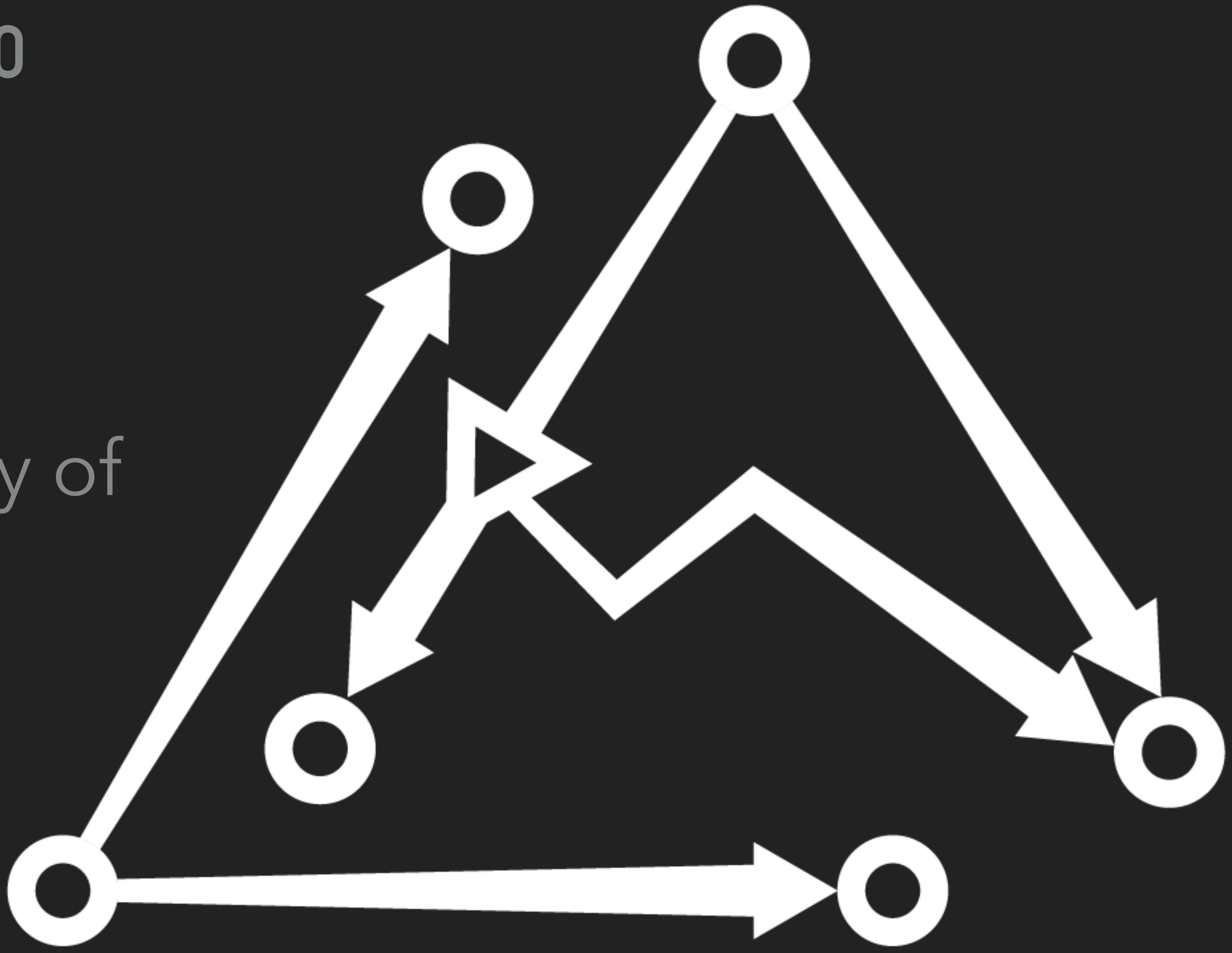
ZKAPS: HOW TO USE PRIVACY PASS FOR PAYMENT-BASED ACCESS TO YOUR APPLICATION

zkSummit 5 - 03/31/20

Anna Kaplan

Least Authority/
Technical University of
Munich

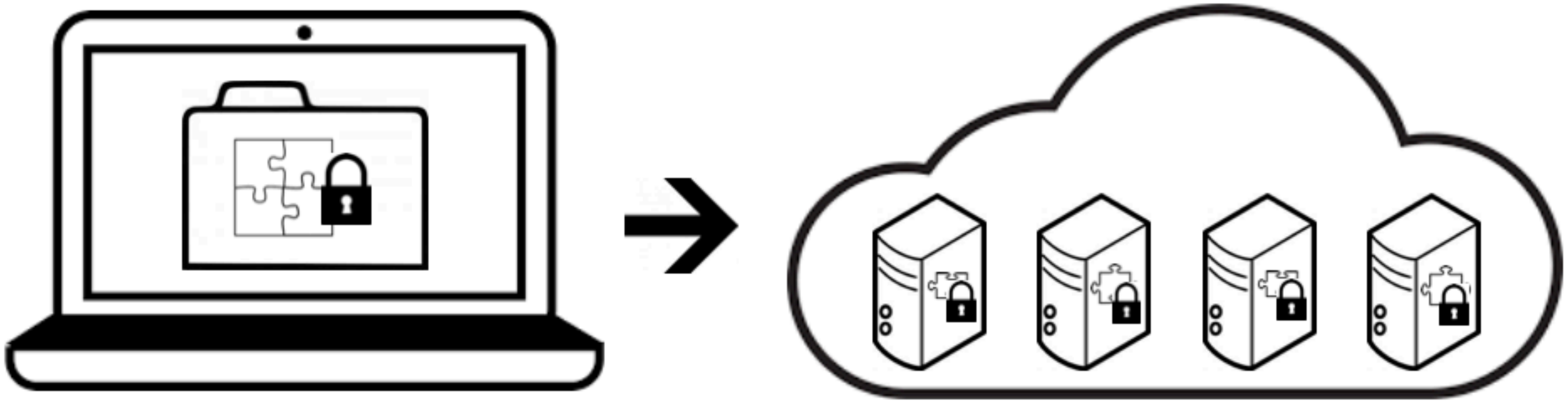


AGENDA

1. The initial **use case**
2. Details about **PrivateStorage**
3. **Privacy Pass** - the original implementation
4. Our use: **Zero Knowledge Access Passes** (ZKAPs)
5. Possibilities for **extensions**

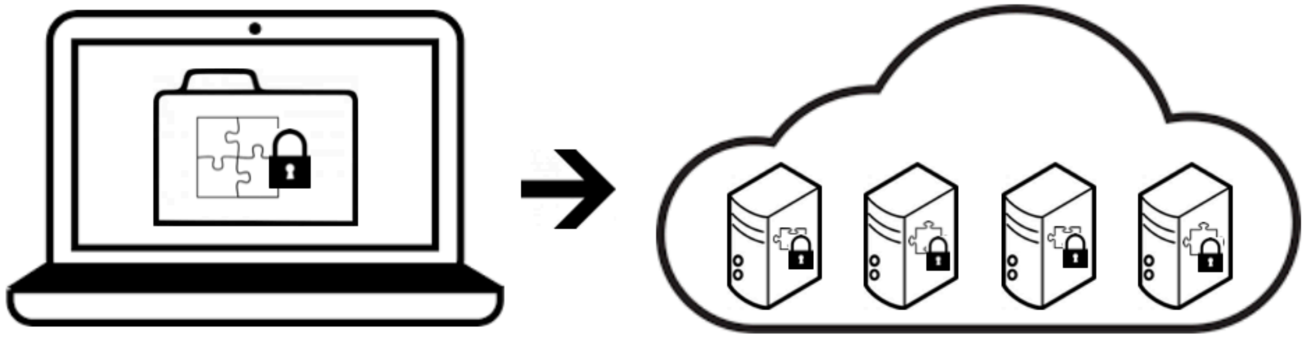
LEAST AUTHORITY BUILT A PRIVATE CLOUD STORAGE SOLUTION

Least Authority
PRIVACY MATTERS



- In 2013: a solution called S4 (Simple Secure Storage Service), based on Least Authority's Tahoe-LAFS, was launched.

LEAST AUTHORITY BUILT A PRIVATE CLOUD STORAGE SOLUTION



Least Authority
PRIVACY MATTERS

What’s it about?

- Client-side encryption
- Sharding of ciphertext
- Potential for decentralised storage servers (grid)
- Not ACL: No user accounts, no passwords, but OCAP: Access based on possession of the capability string

OUR PROBLEM: FIAT CURRENCY PAYMENT PROCESSING

- Name
- Email address
- Location (for VAT)
- Transaction Data

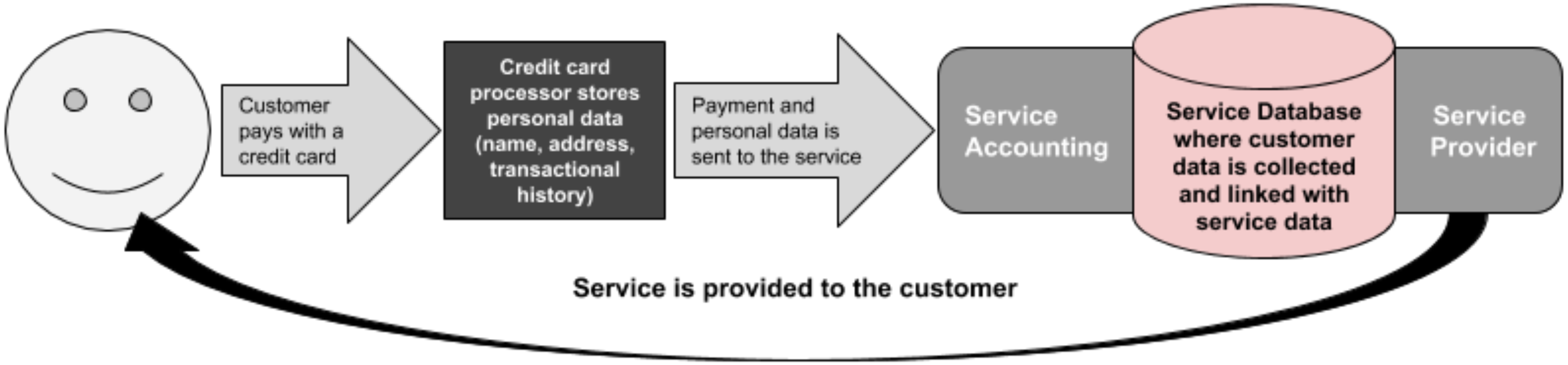
…and sharing with these other companies

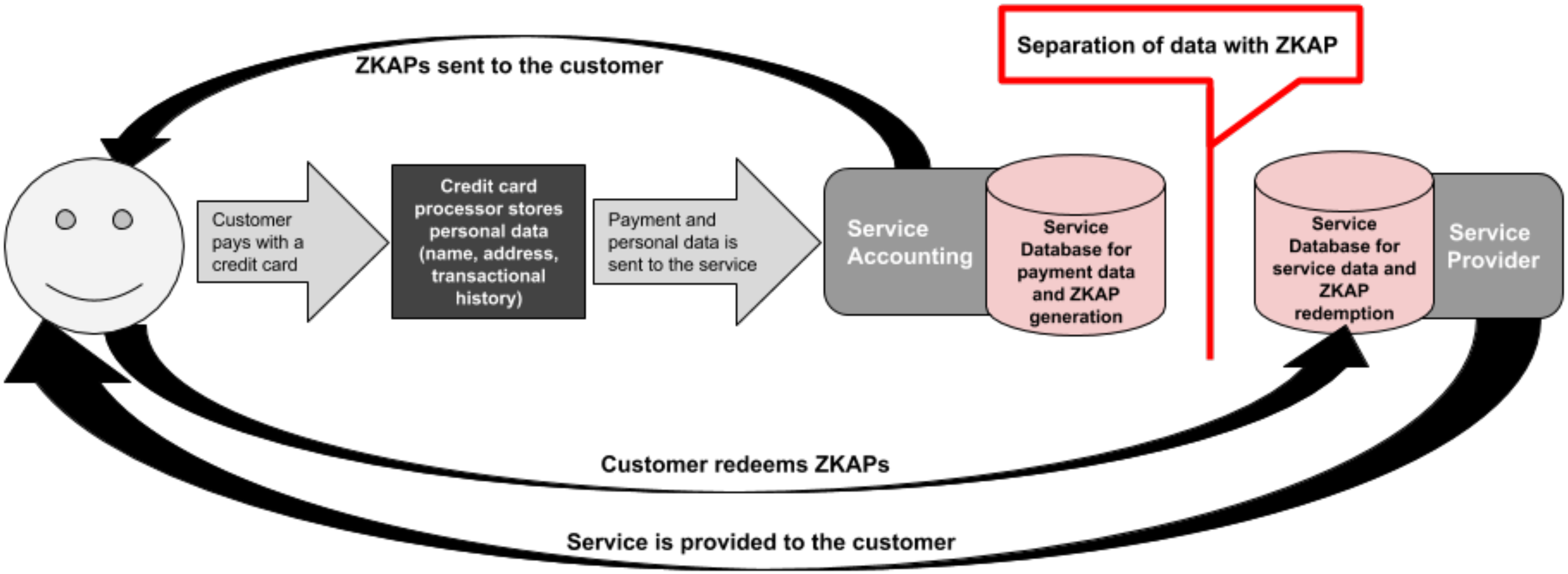
stripe

chargebee

= collecting personal data just for payments

HOW TO SEPARATE SERVICE ACCOUNTING AND PROVIDER?


Customer pays with a credit card
Credit card processor stores personal data (name, address, transactional history)
Payment and personal data is sent to the service
Service Accounting
Service Database where customer data is collected and linked with service data
Service Provider
Service is provided to the customer


ZKAPs sent to the customer
Separation of data with ZKAP
Customer pays with a credit card
Credit card processor stores personal data (name, address, transactional history)
Payment and personal data is sent to the service
Service Accounting
Service Database for payment data and ZKAP generation
Service Database for service data and ZKAP redemption
Service Provider
Customer redeems ZKAPs
Service is provided to the customer

FROM S4 (SIMPLE SECURE STORAGE SERVICE) TO PRIVATE STORAGE

Least Authority
PRIVACY MATTERS

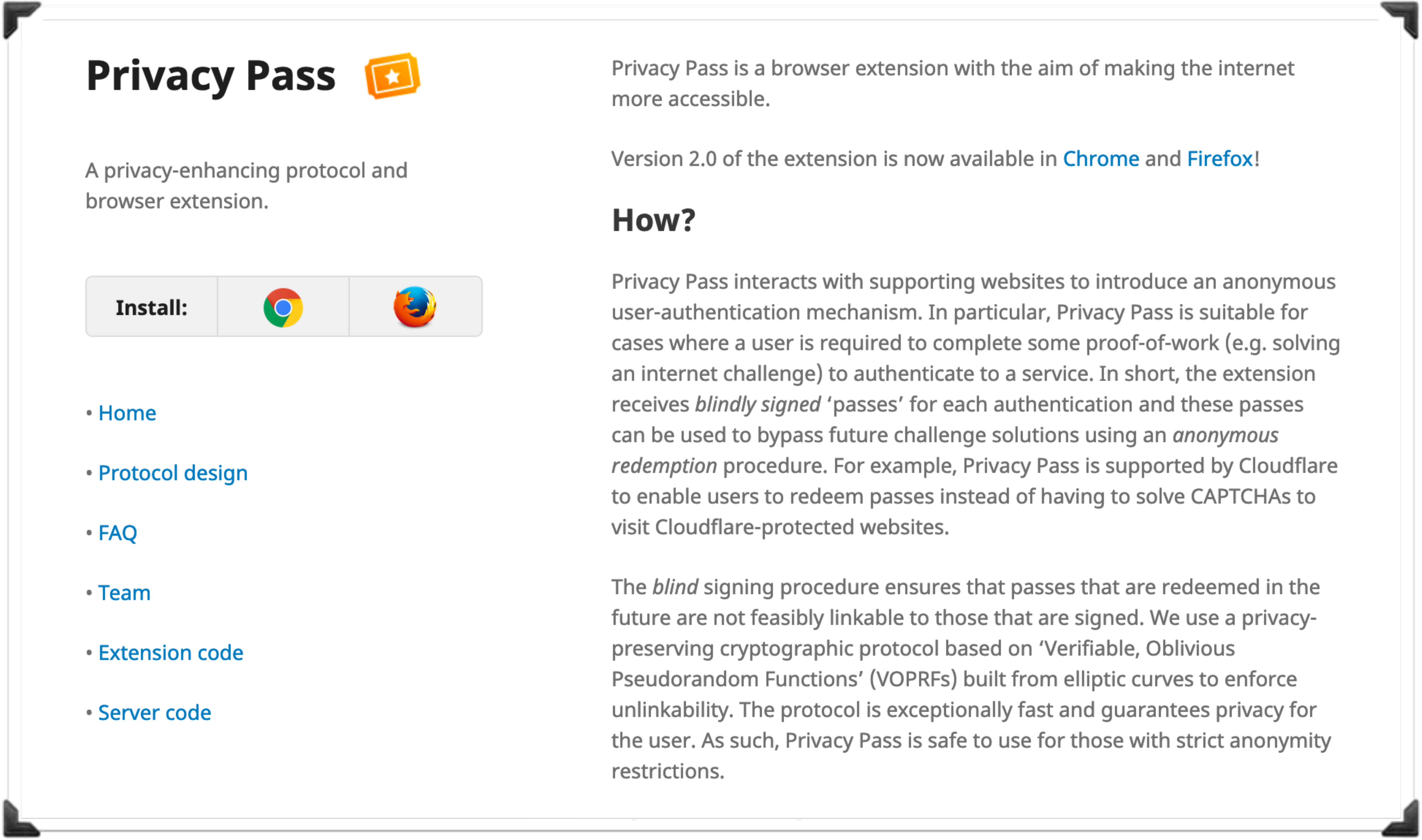
PRIVATE
STORAGE

- Least Authority and Private Internet Access (privacy focused VPN provider) announce **PrivateStorage**
- PrivacyStorage is private, secure and end-to-end encrypted cloud storage solution, based on Least Authority's Tahoe-LAFS and developed from S4
- Private Storage therefore implements **Zero Knowledge Access Passes (ZKAPs)** as a variation of Privacy Pass

ZERO KNOWLEDGE ACCESS PASSES (ZKAPS)

PRIVACY PASS

Work by Alex Davidson, Ian Goldberg, Nick Sullivan, George Tankersley, and Filippo Valsorda, 2018
https://privacypass.github.io/



Privacy Pass

A privacy-enhancing protocol and browser extension.

Install:

- Home
- Protocol design
- FAQ
- Team
- Extension code
- Server code

Privacy Pass is a browser extension with the aim of making the internet more accessible.

Version 2.0 of the extension is now available in Chrome and Firefox!

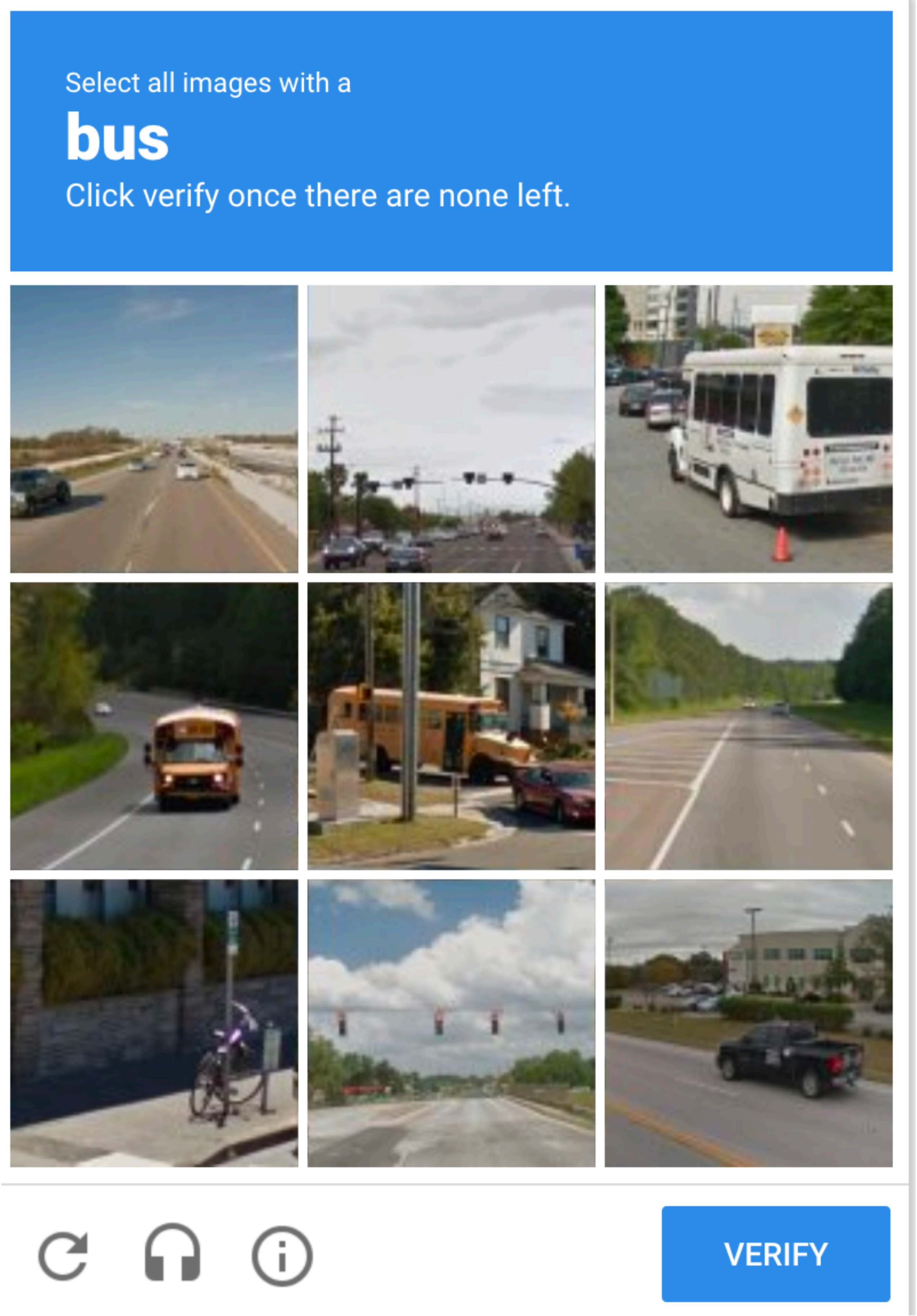
How?

Privacy Pass interacts with supporting websites to introduce an anonymous user-authentication mechanism. In particular, Privacy Pass is suitable for cases where a user is required to complete some proof-of-work (e.g. solving an internet challenge) to authenticate to a service. In short, the extension receives *blindly signed* 'passes' for each authentication and these passes can be used to bypass future challenge solutions using an *anonymous redemption* procedure. For example, Privacy Pass is supported by Cloudflare to enable users to redeem passes instead of having to solve CAPTCHAs to visit Cloudflare-protected websites.

The *blind* signing procedure ensures that passes that are redeemed in the future are not feasibly linkable to those that are signed. We use a privacy-preserving cryptographic protocol based on 'Verifiable, Oblivious Pseudorandom Functions' (VOPRFs) built from elliptic curves to enforce unlinkability. The protocol is exceptionally fast and guarantees privacy for the user. As such, Privacy Pass is safe to use for those with strict anonymity restrictions.

PRIVACY PASS – MOTIVATION

- Developed by CLOUDFLARE
- Cloudflare needs to prevent malicious attacks, e.g. comment spam or SQL attacks, from the web
- Cloudflare does this through **IP reputation** assessment
- How to know that’s a “good" IP address? I have a great solution for you!

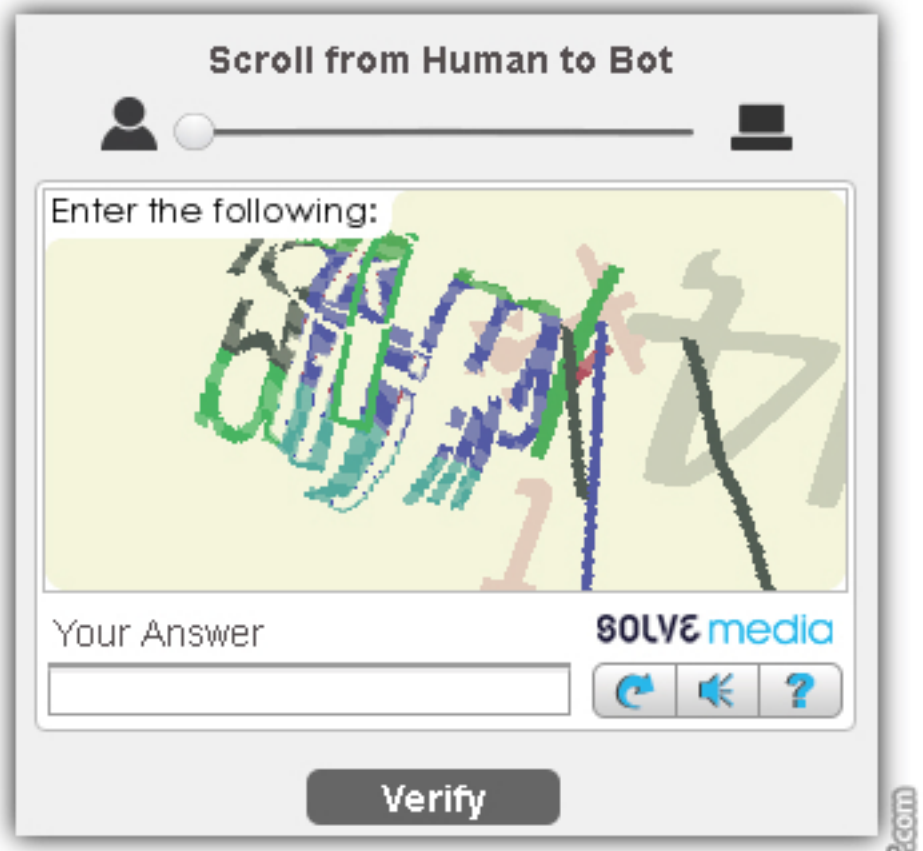


Select all images with a

bus

Click verify once there are none left.

Screenshot by me,
10/29/2019



Why can't I read that?
Am I a robot?

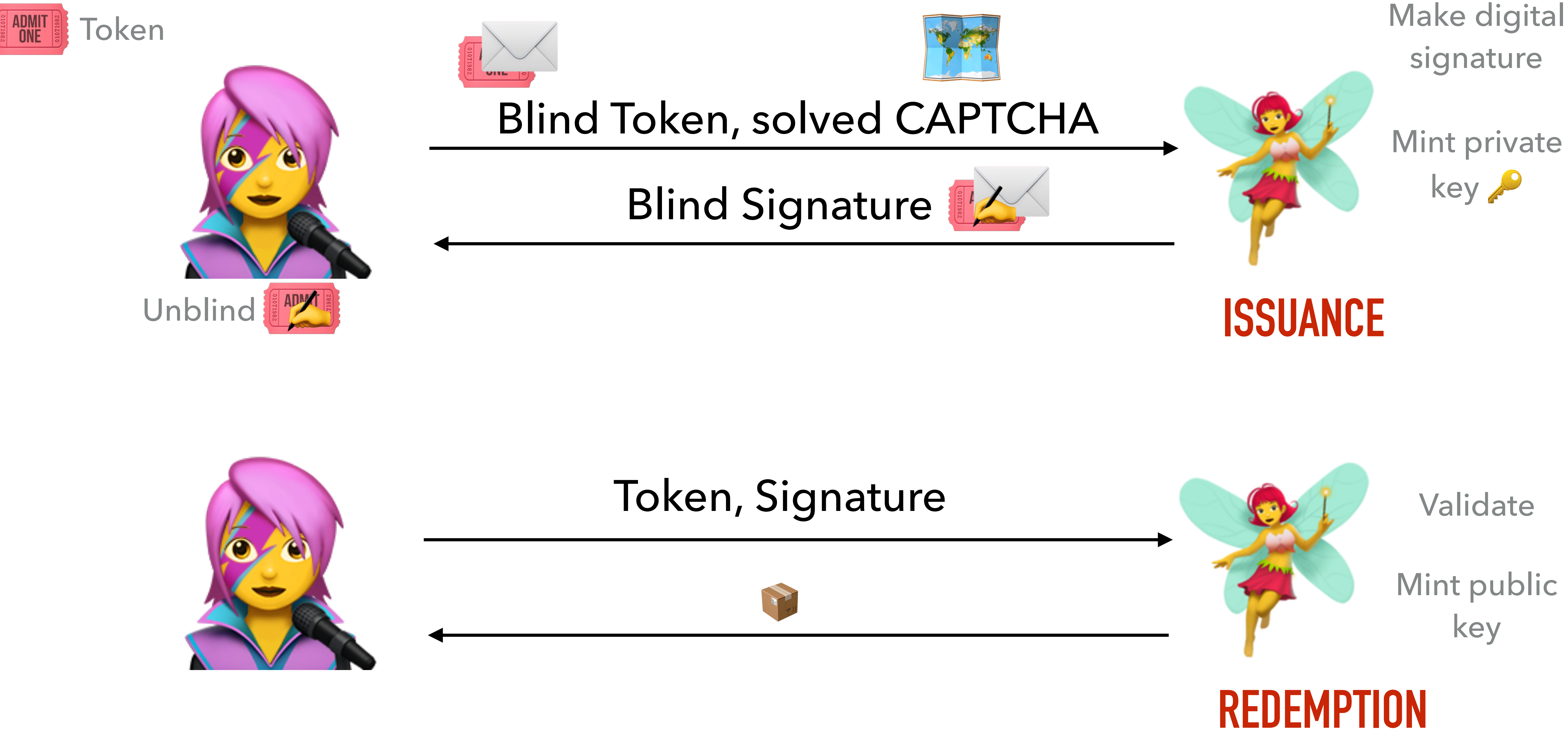
PRIVACY PASS – BACKGROUND

- Idea based on **Ecash (Chaum 1983)**:
    - You take a token, blind it, get a blind signature
    - Issuance and Redemption are unlinkable
- After Real World Crypto 2016: How to apply the idea of **blinded signatures to not always having to solve CAPTCHAs**?
    - Filippo Valsorda and George Tankersley came up with first specification for a blinded token to be issued when a CAPTCHA is solved, and can be redeemed later
    - Take a token, blind it, send it to Cloudflare with CAPTCHA solution, get a blind signature in response, which you can later redeem
    - These are unlinkable for Cloudflare

PRIVACY PASS – BACKGROUND

- Problem: Ecash was based on RSA. 1980s cryptography is slow!
- At PETS 2016, Davidson, Tankersley, and Valsorda asked for help and Dan Boneh mentioned EC-OPRFs.
- OPRF: Oblivious Pseudo-Random Function
- **Batched Elliptic Curve VOPRF with redemption** (Tankersley)
    - Multiple simultaneous OPRFs based on Elliptic Curve multiplication
    - VRF-like public verification
    - Batched validation for more efficiency
- **VOPRFs** 🆚 **Ecash:** Ecash is publicly verifiable 🆚 VOPRFs only verifiable in the redemption phase by the issuer

IDEA: “MODERNIZED ECASH” WITH NO CASH INVOLVED



🎟️ Token

Blind Token, solved CAPTCHA

Blind Signature

Unblind

Make digital signature

Mint private key 🔑

ISSUANCE

Token, Signature

Validate

Mint public key

REDEMPTION

WHERE DO ZERO-KNOWLEDGE PROOFS COME INTO PLAY?

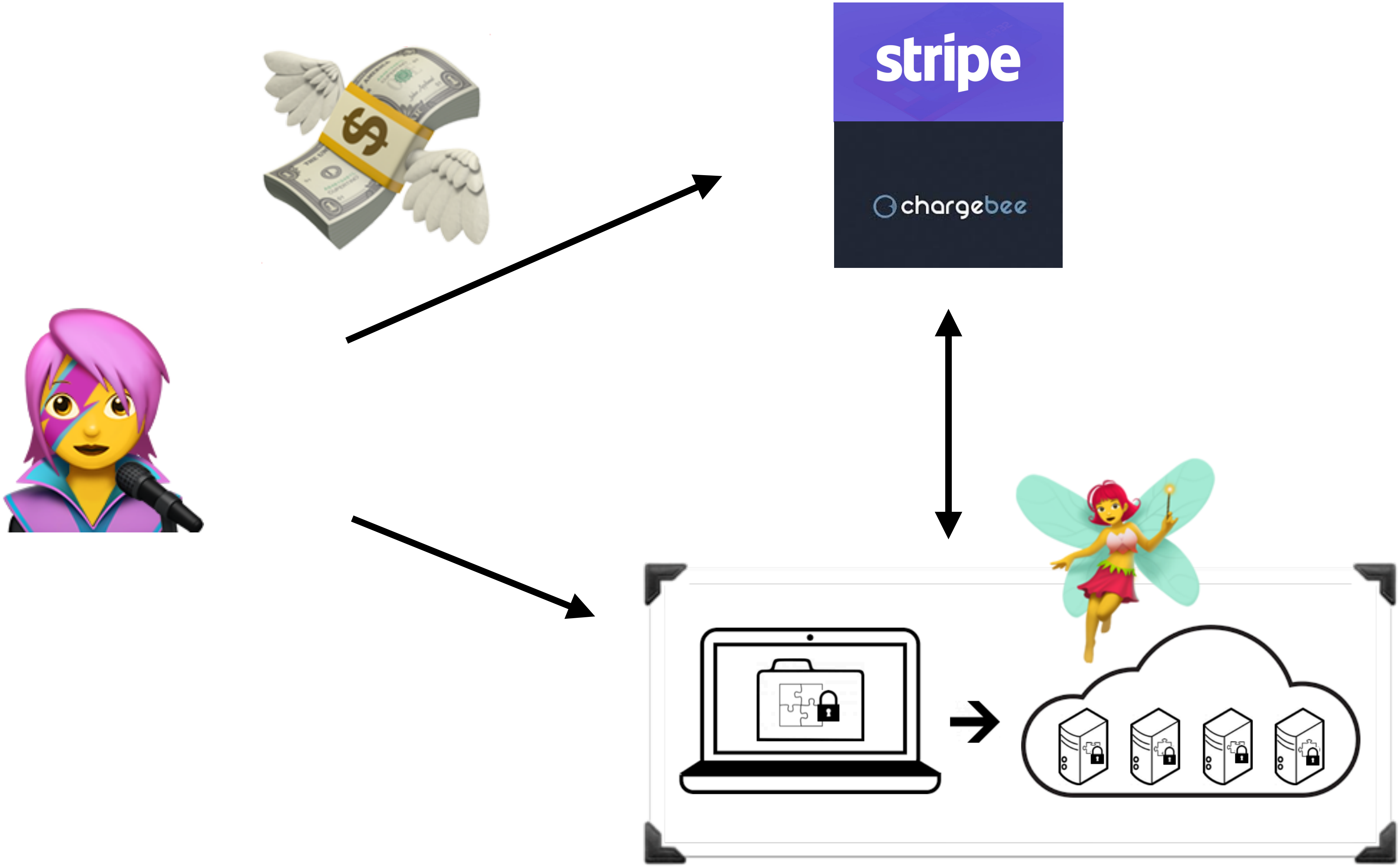
EC-VOPRFs use a **Discrete Log Equivalence Proof**

- Short ZKP that two pairs of points have the same Discrete Log, denounced **DLEQ(P:R == Q:S)**.

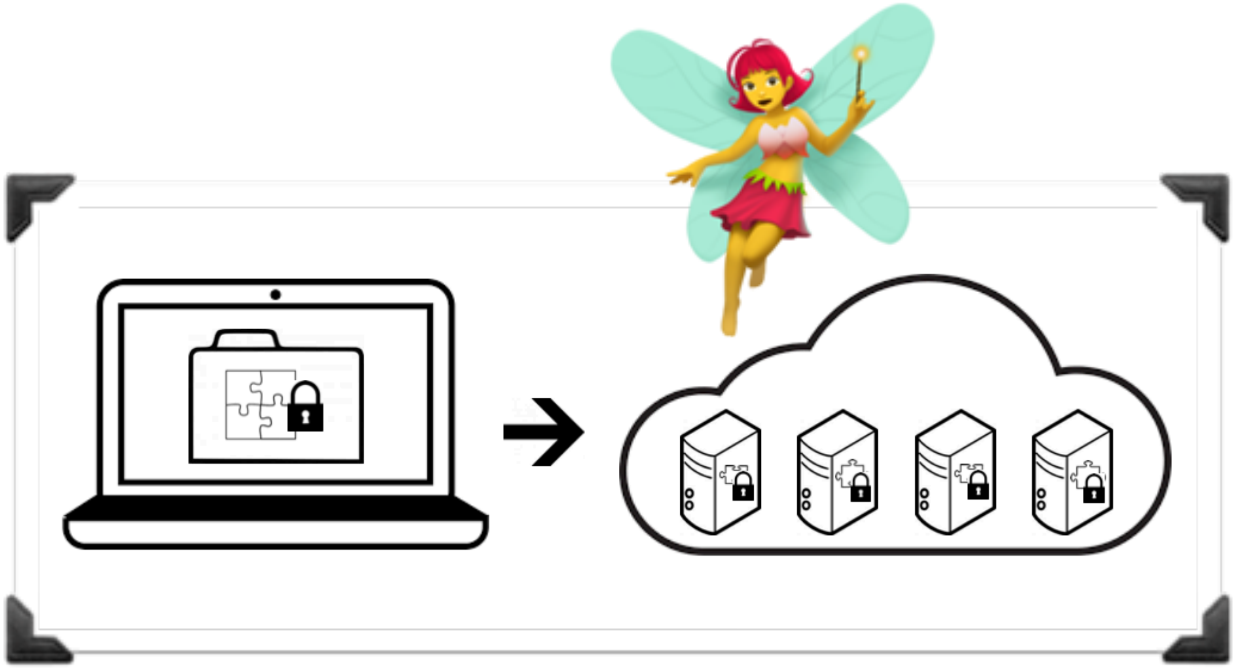
CURRENT STATE AND OTHER IDEAS TO THINK ABOUT

- Privacy Pass exists as an extension for Firefox or Chrome
- **Other ideas** to use this idea:
  - Anonymous session resumption for TLS
  - Anonymous referral code mechanism (e.g. discount codes) - used in Brave browser for ads
  - Single bit ZKP (e.g. Am I over 18?)

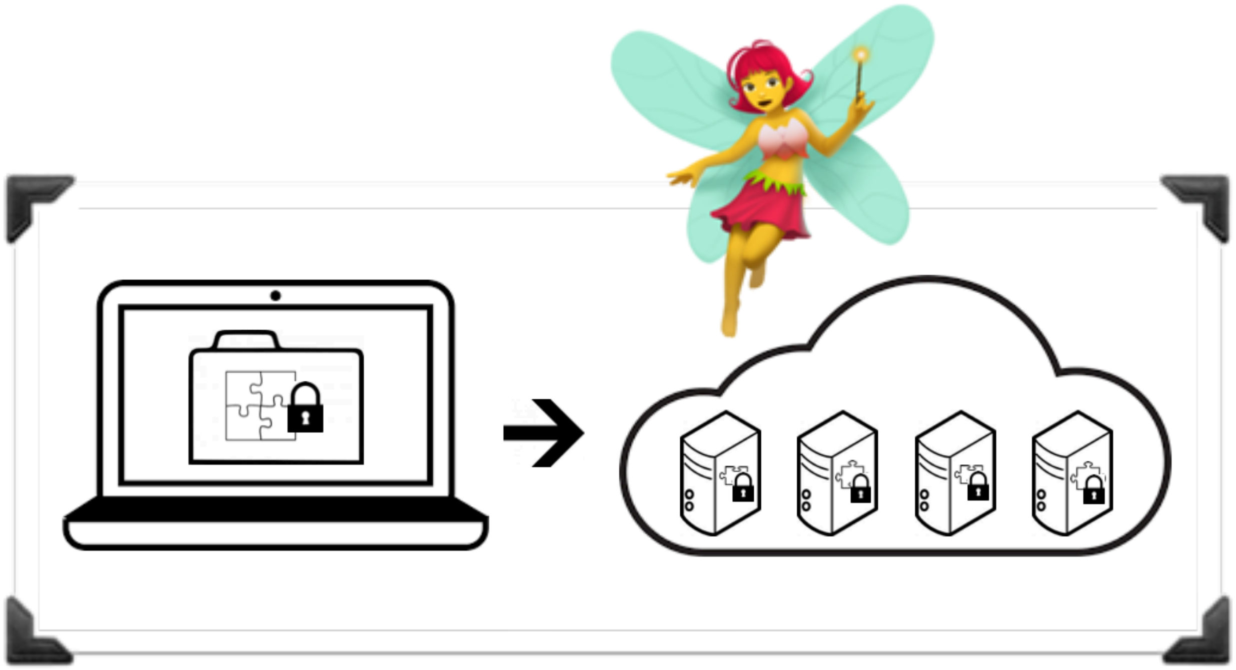
OUR USE: ZERO KNOWLEDGE ACCESS PASSES (ZKAPS)



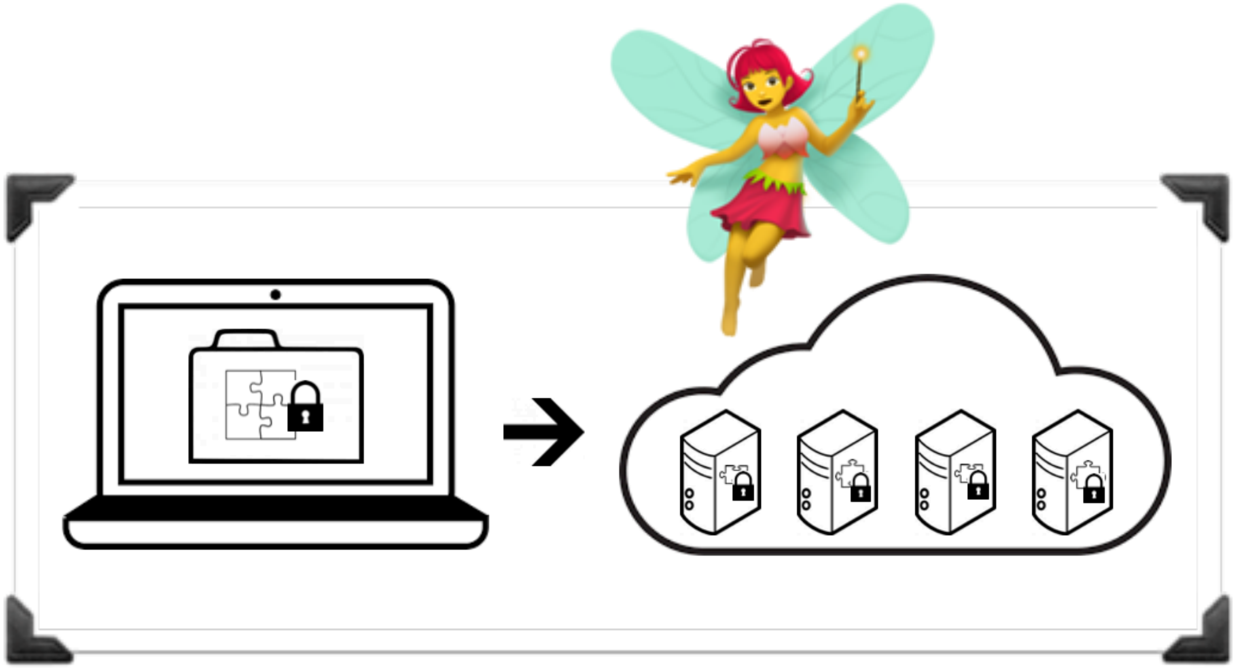
OUR USE: ZERO KNOWLEDGE ACCESS PASSES (ZKAPS)



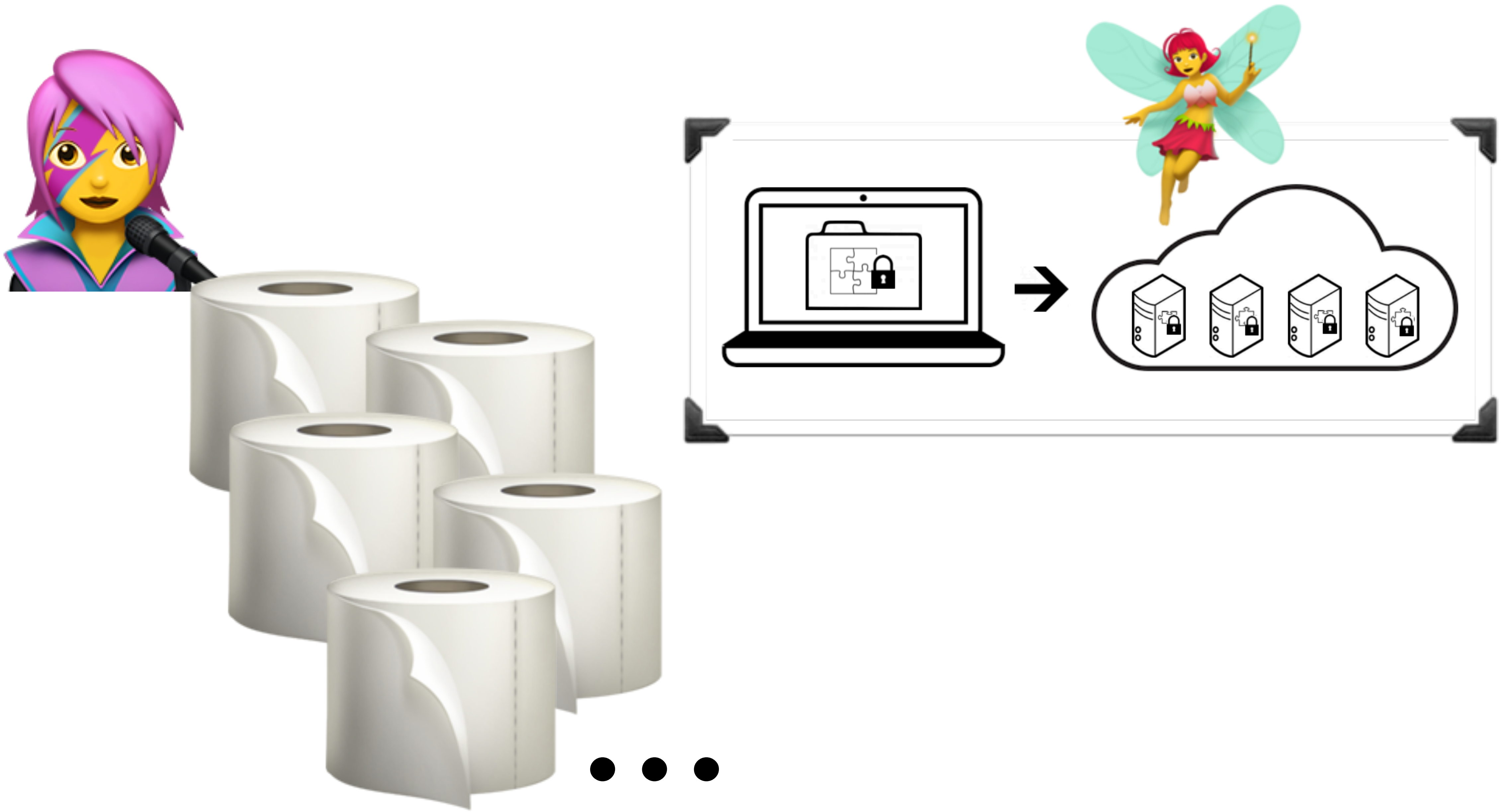
OUR USE: ZERO KNOWLEDGE ACCESS PASSES (ZKAPS)



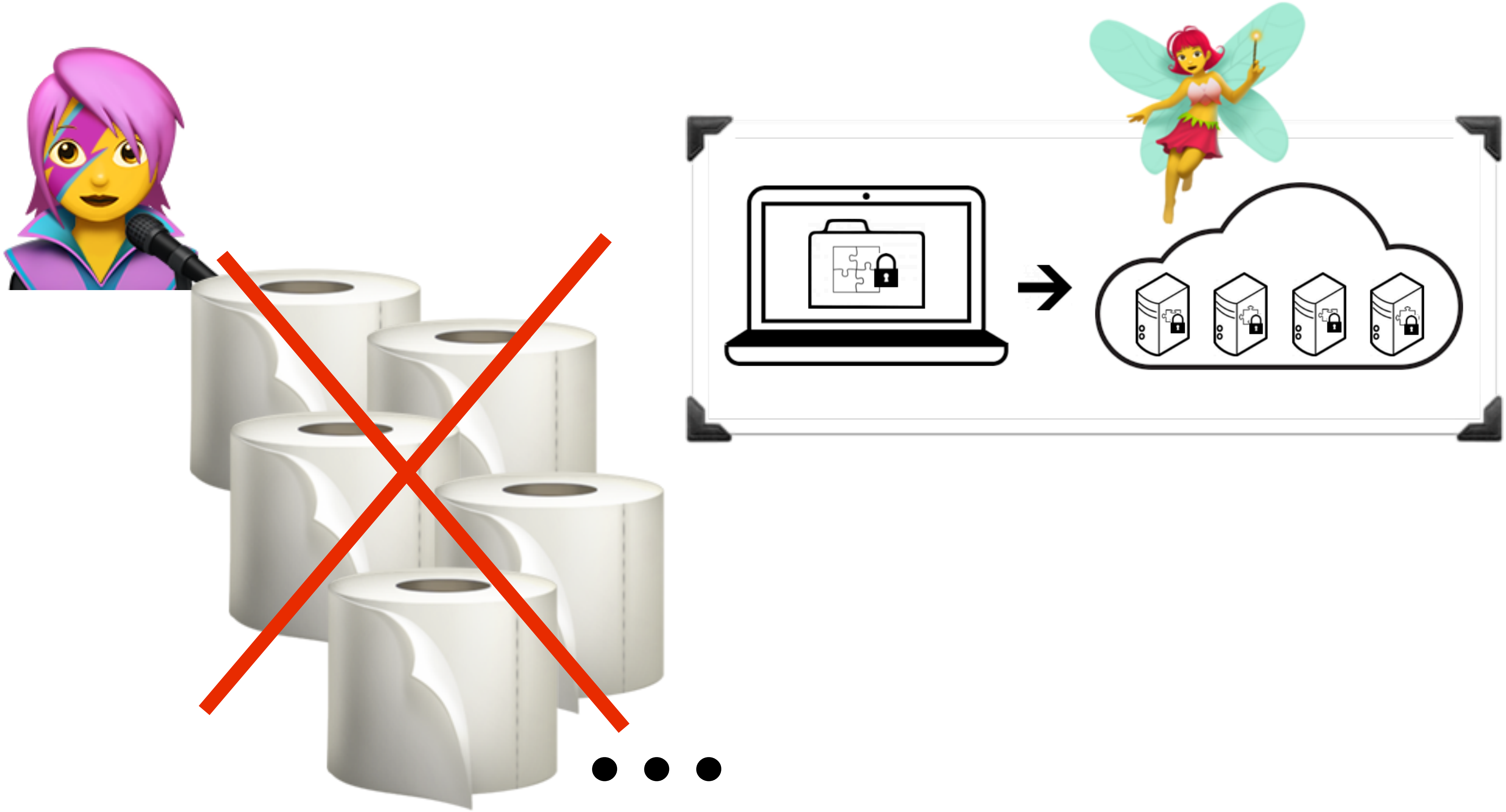
OUR USE: ZERO KNOWLEDGE ACCESS PASSES (ZKAPS)



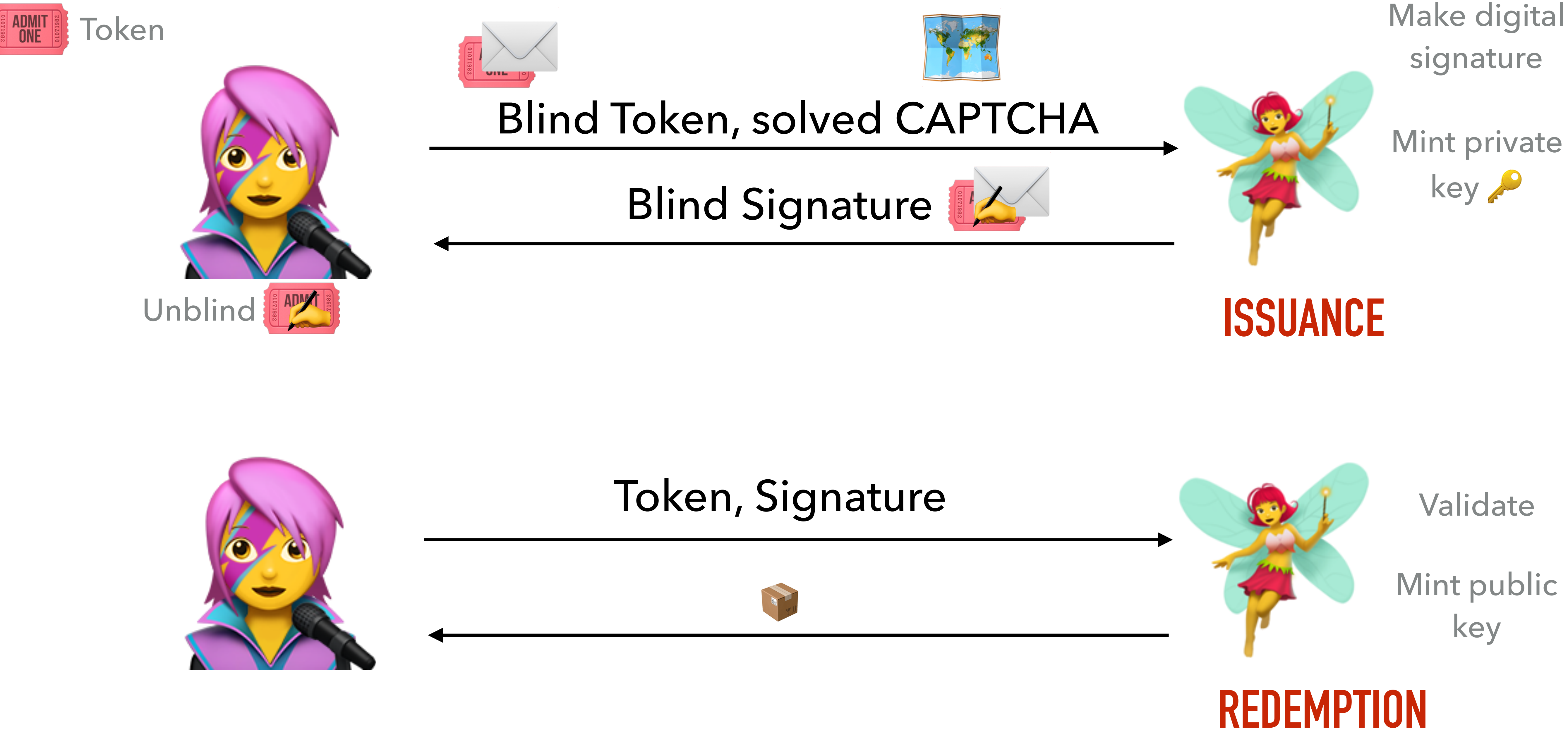
OUR USE: ZERO KNOWLEDGE ACCESS PASSES (ZKAPS)



OUR USE: ZERO KNOWLEDGE ACCESS PASSES (ZKAPS)



OUR USE: ZERO KNOWLEDGE ACCESS PASSES (ZKAPS)



🎟️ Token

Blind Token, solved CAPTCHA

Blind Signature

Unblind

Make digital signature

Mint private key 🔑

ISSUANCE

Token, Signature

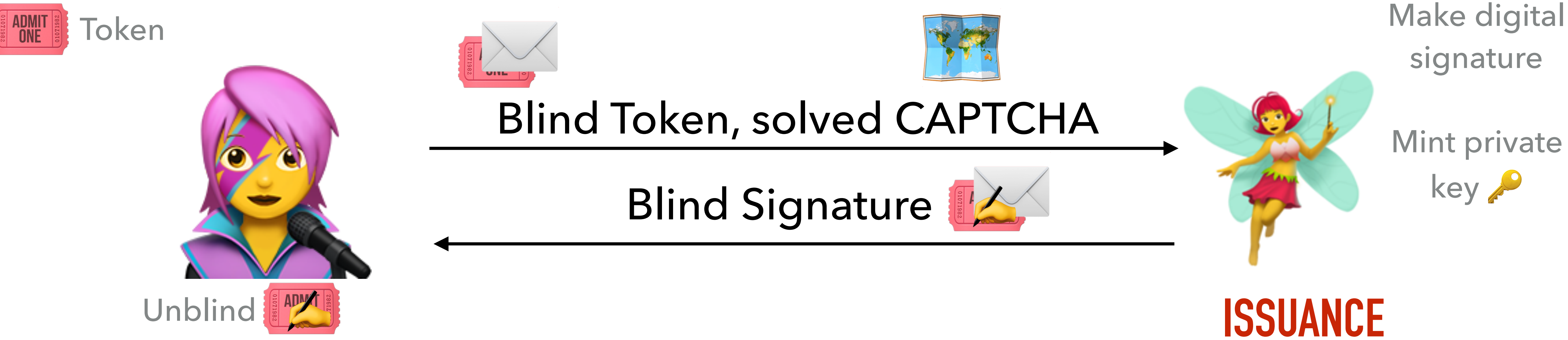
📦

Validate

Mint public key

REDEMPTION

OUR USE: ZERO KNOWLEDGE ACCESS PASSES (ZKAPS)



🎟 Token

Blind Token, solved CAPTCHA

Blind Signature

Unblind

Make digital signature

Mint private key 🔑

ISSUANCE

H: Hash function

OUR USE: ZERO KNOWLEDGE ACCESS PASSES (ZKAPS)

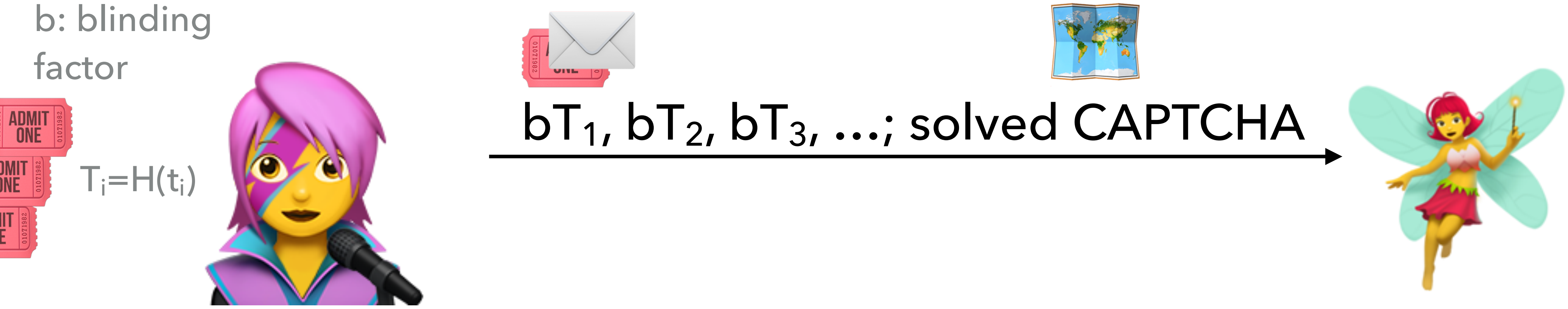
ISSUANCE

H: Hash function

OUR USE: ZERO KNOWLEDGE ACCESS PASSES (ZKAPS)

$t_1, t_2, t_3, \ldots$

b: blinding factor



$T_i = H(t_i)$

$bT_1, bT_2, bT_3, \ldots$; solved CAPTCHA

ISSUANCE

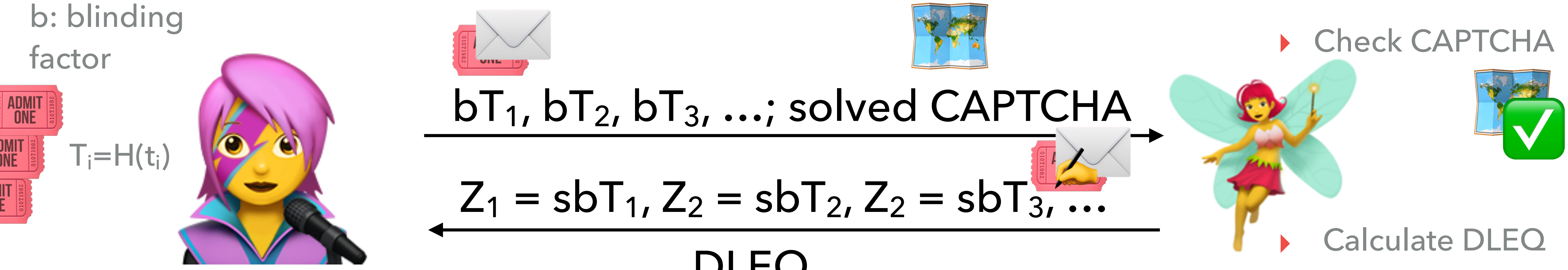
H: Hash function

OUR USE: ZERO KNOWLEDGE ACCESS PASSES (ZKAPS)

$t_1, t_2, t_3, \ldots$

b: blinding factor

$T_i=H(t_i)$



$bT_1, bT_2, bT_3, \ldots$; solved CAPTCHA

$Z_1 = sbT_1, Z_2 = sbT_2, Z_2 = sbT_3, \ldots$

DLEQ

- Check CAPTCHA
- Calculate DLEQ

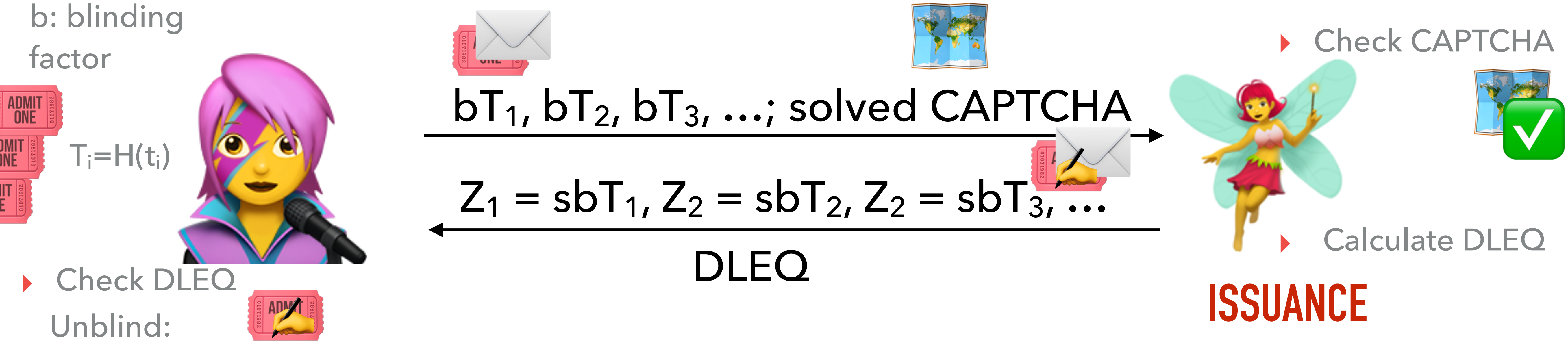
ISSUANCE

H: Hash function

OUR USE: ZERO KNOWLEDGE ACCESS PASSES (ZKAPS)

$t_1, t_2, t_3, \ldots$

b: blinding factor



$T_i=H(t_i)$

$bT_1, bT_2, bT_3, \ldots$; solved CAPTCHA

$Z_1 = sbT_1, Z_2 = sbT_2, Z_2 = sbT_3, \ldots$

DLEQ

- Check CAPTCHA
- Calculate DLEQ

ISSUANCE

- Check DLEQ

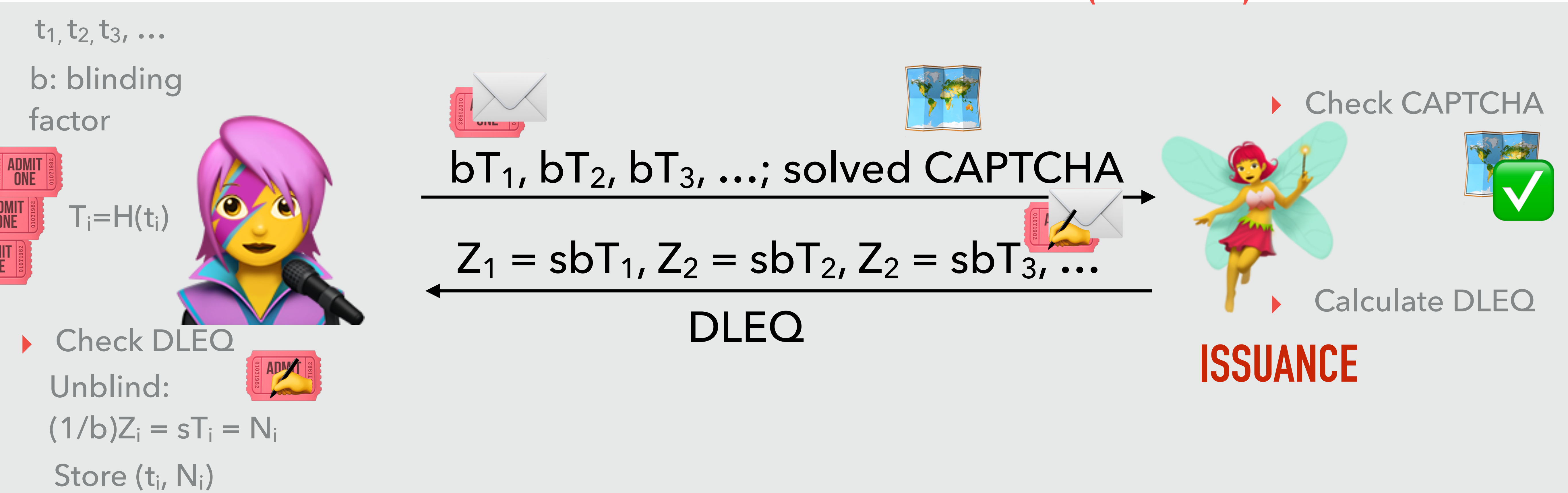
Unblind:

$(1/b)Z_i = sT_i = N_i$

Store (t_i, N_i)

H: Hash function

OUR USE: ZERO KNOWLEDGE ACCESS PASSES (ZKAPS)



t_1, t_2, t_3, …

b: blinding factor

$T_i=H(t_i)$

bT_1, bT_2, bT_3, …; solved CAPTCHA

$Z_1 = sbT_1, Z_2 = sbT_2, Z_2 = sbT_3$, …

DLEQ

- Check CAPTCHA
- Calculate DLEQ

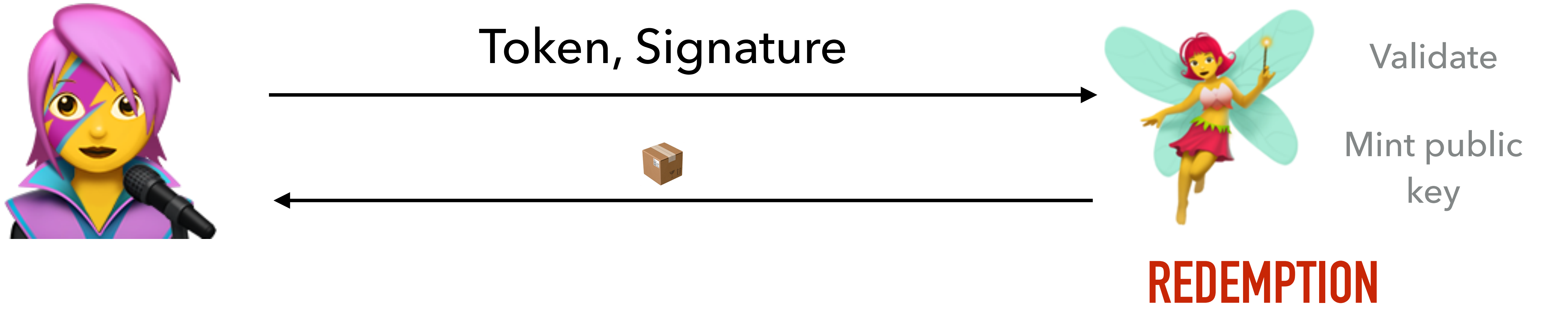
ISSUANCE

- Check DLEQ

Unblind:

$(1/b)Z_i = sT_i = N_i$

Store (t_i, N_i)



Token, Signature

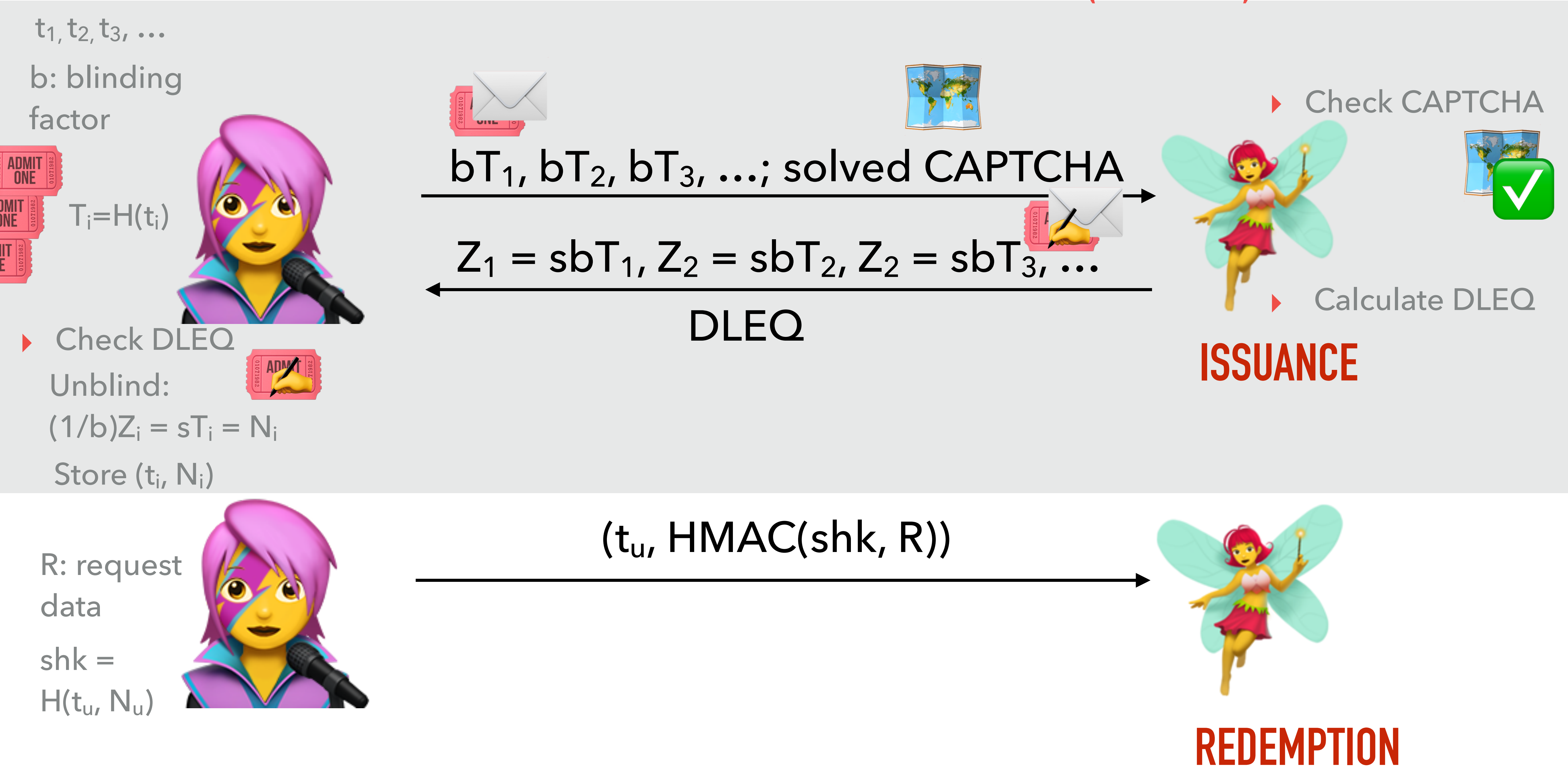
Validate

Mint public key

REDEMPTION

H: Hash function

OUR USE: ZERO KNOWLEDGE ACCESS PASSES (ZKAPS)



$t_1, t_2, t_3, \ldots$

b: blinding factor

$T_i = H(t_i)$

- Check DLEQ

Unblind:

$(1/b)Z_i = sT_i = N_i$

Store (t_i, N_i)

$bT_1, bT_2, bT_3, \ldots$; solved CAPTCHA

$Z_1 = sbT_1, Z_2 = sbT_2, Z_2 = sbT_3, \ldots$

DLEQ

- Check CAPTCHA
- Calculate DLEQ

ISSUANCE

R: request data

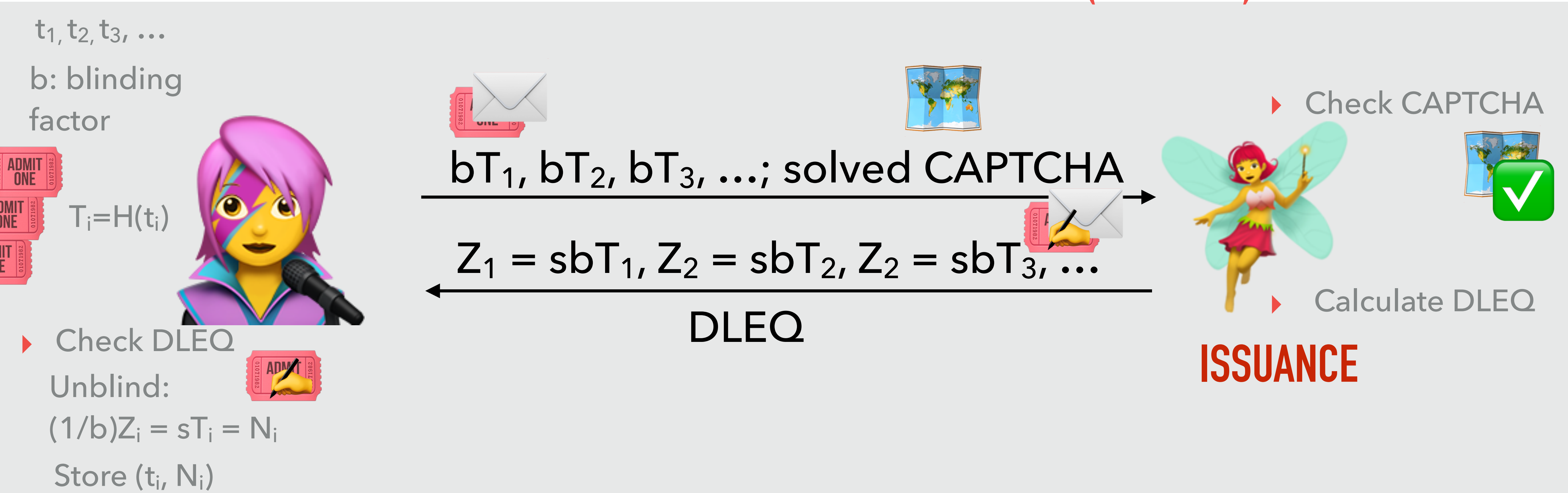
shk = $H(t_u, N_u)$

$(t_u, \text{HMAC}(shk, R))$

REDEMPTION

H: Hash function

OUR USE: ZERO KNOWLEDGE ACCESS PASSES (ZKAPS)



$t_{1,} t_{2,} t_3$, …

b: blinding factor

T_i=H(t_i)

- Check DLEQ

Unblind:

(1/b)Z_i = sT_i = N_i

Store (t_i, N_i)

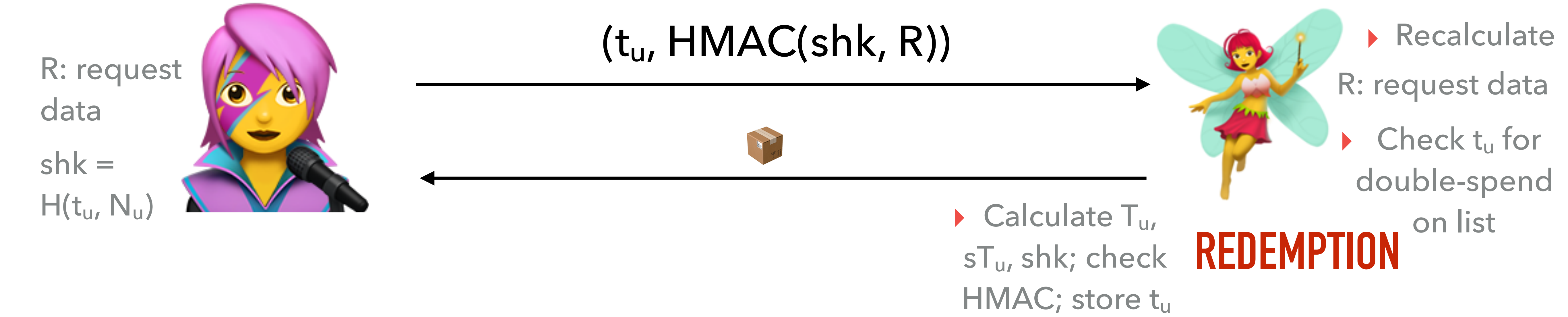
bT_1, bT_2, bT_3, …; solved CAPTCHA

Z_1 = sbT_1, Z_2 = sbT_2, Z_2 = sbT_3, …

DLEQ

- Check CAPTCHA
- Calculate DLEQ

ISSUANCE



R: request data

shk = H(t_u, N_u)

(t_u, HMAC(shk, R))

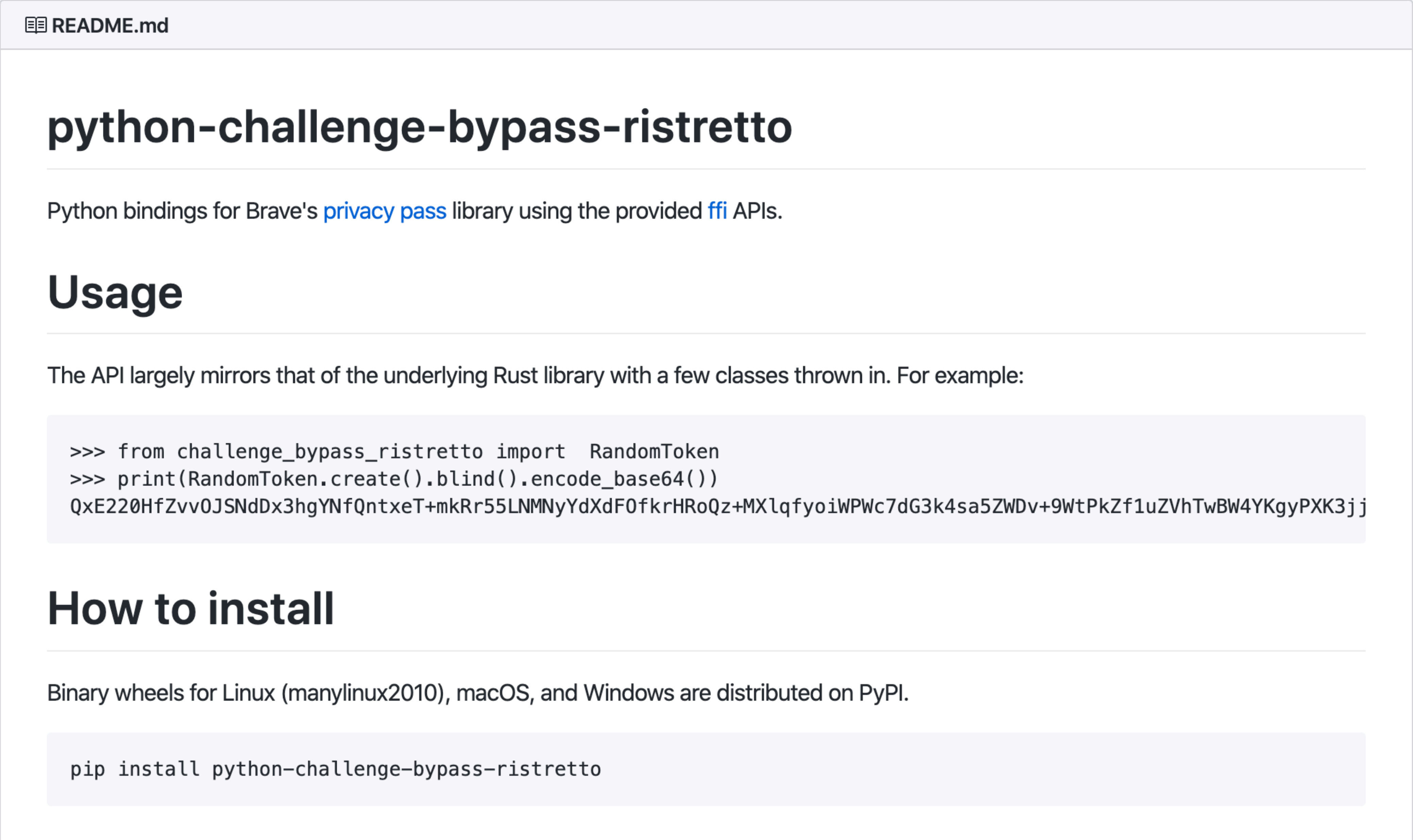
- Calculate T_u, sT_u, shk; check HMAC; store t_u
- Recalculate R: request data
- Check t_u for double-spend on list

REDEMPTION

*** this is not the full specification ***

OUR USE: ZERO KNOWLEDGE ACCESS PASSES (ZKAPS)

https://github.com/LeastAuthority/python-challenge-bypass-ristretto



README.md

python-challenge-bypass-ristretto

Python bindings for Brave's privacy pass library using the provided ffi APIs.

Usage

The API largely mirrors that of the underlying Rust library with a few classes thrown in. For example:

```
>>> from challenge_bypass_ristretto import  RandomToken
>>> print(RandomToken.create().blind().encode_base64())
QxE220HfZvvOJSNdDx3hgYNfQntxeT+mkRr55LNMNyYdXdFOfkrHRoQz+MXlqfyoiWPWc7dG3k4sa5ZWDv+9WtPkZf1uZVhTwBW4YKgyPXK3jj
```

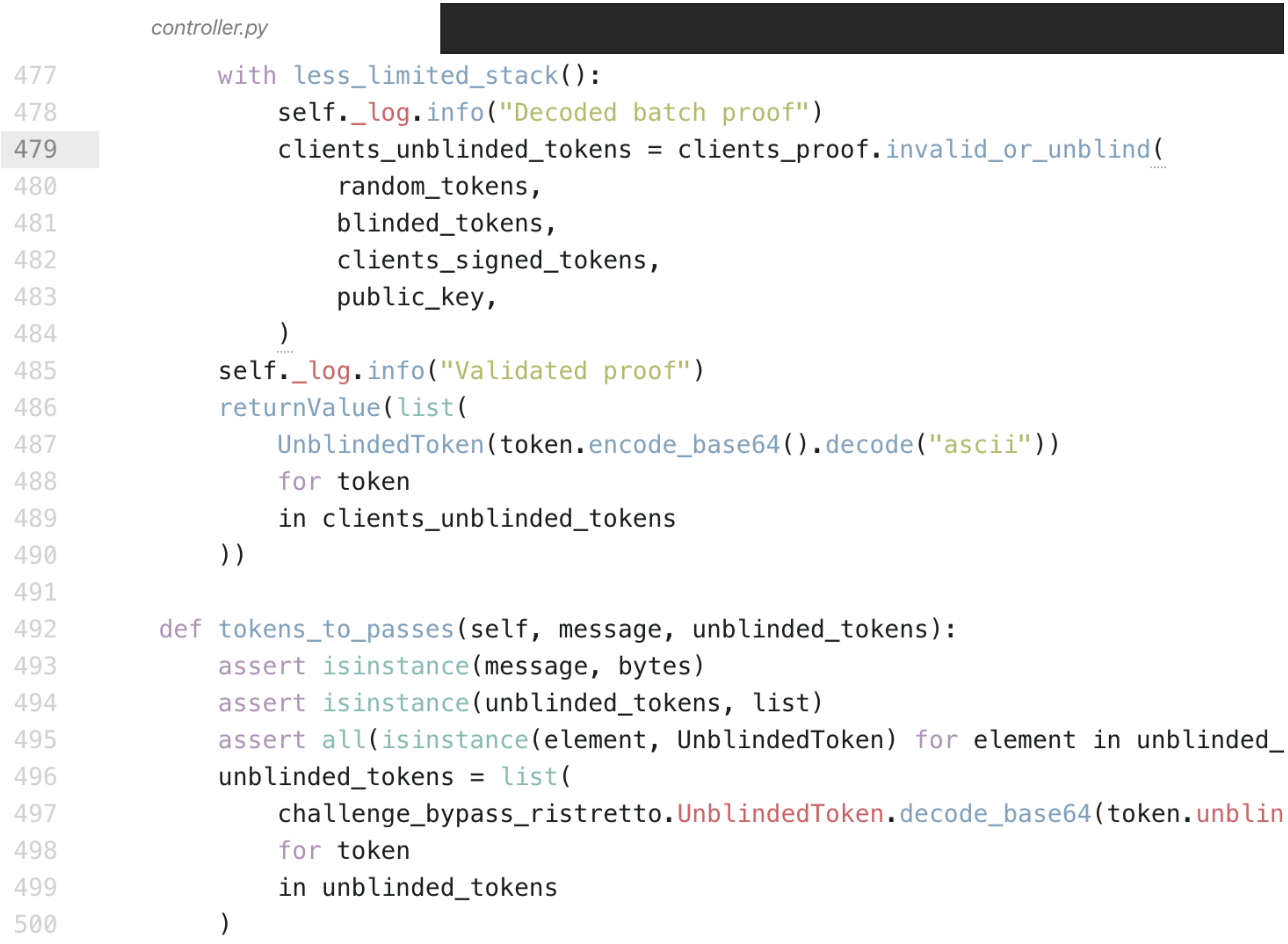
How to install

Binary wheels for Linux (manylinux2010), macOS, and Windows are distributed on PyPI.

```
pip install python-challenge-bypass-ristretto
```

OUR USE: ZERO KNOWLEDGE ACCESS PASSES (ZKAPS)

https://github.com/PrivateStorageio/ZKAPAuthorizer/blob/master/src/_zkapauthorizer/controller.py#L479



controller.py [redacted]

```
477         with less_limited_stack():
478             self._log.info("Decoded batch proof")
479             clients_unblinded_tokens = clients_proof.invalid_or_unblind(
480                 random_tokens,
481                 blinded_tokens,
482                 clients_signed_tokens,
483                 public_key,
484             )
485         self._log.info("Validated proof")
486         returnValue(list(
487             UnblindedToken(token.encode_base64().decode("ascii"))
488             for token
489             in clients_unblinded_tokens
490         ))
491 
492     def tokens_to_passes(self, message, unblinded_tokens):
493         assert isinstance(message, bytes)
494         assert isinstance(unblinded_tokens, list)
495         assert all(isinstance(element, UnblindedToken) for element in unblinded_
496         unblinded_tokens = list(
497             challenge_bypass_ristretto.UnblindedToken.decode_base64(token.unblin
498             for token
499             in unblinded_tokens
500         )
```

ZERO KNOWLEDGE ACCESS PASSES (ZKAPS)

USE UNLINKABLE ACCESS PASSES FOR YOUR USE CASE!

LINKS AND REFERENCES

- https://privacypass.github.io/
- https://privacypass.github.io/protocol/
- https://github.com/brave-intl/challenge-bypass-ristretto
- https://github.com/LeastAuthority/python-challenge-bypass-ristretto
- https://github.com/PrivateStorageio/ZKAPAuthorizer
- https://leastauthority.com/blog/the-path-from-s4-to-privatestorage/

Least Authority

PRIVACY MATTERS

https://leastauthority.com

Anna@LeastAuthority.com

Twitter:
@Kaplannie
@LeastAuthority

EVERY PROGRAM AND EVERY PRIVILEGED USER OF THE SYSTEM SHOULD OPERATE USING THE LEAST AMOUNT OF PRIVILEGE NECESSARY TO COMPLETE THE JOB.

Jerome Saltzer