CONIKS (KEY TRANSPARENCY)

Slides adapted from Marcela Melara

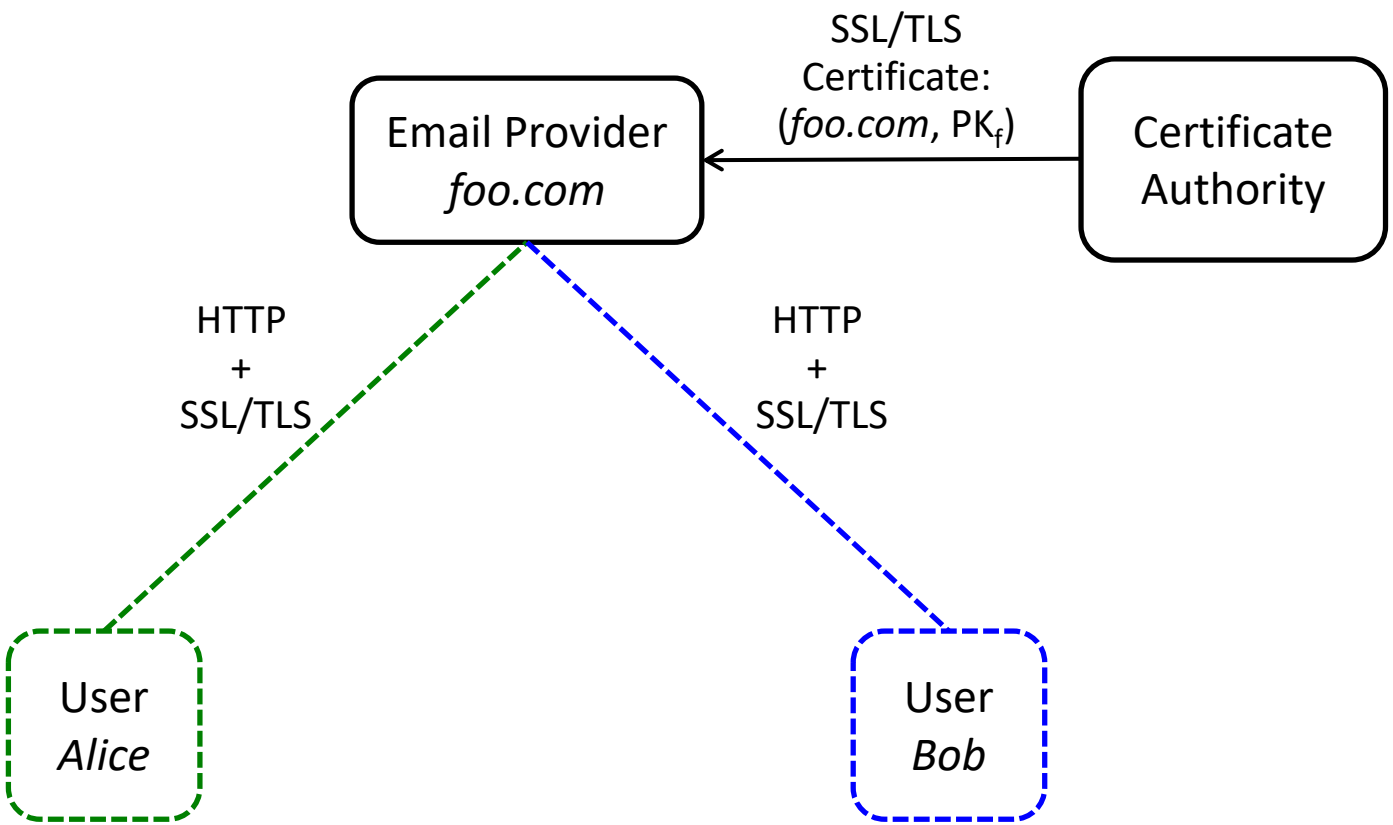
Any scriber for today?

- Sign up for presentations or scribers

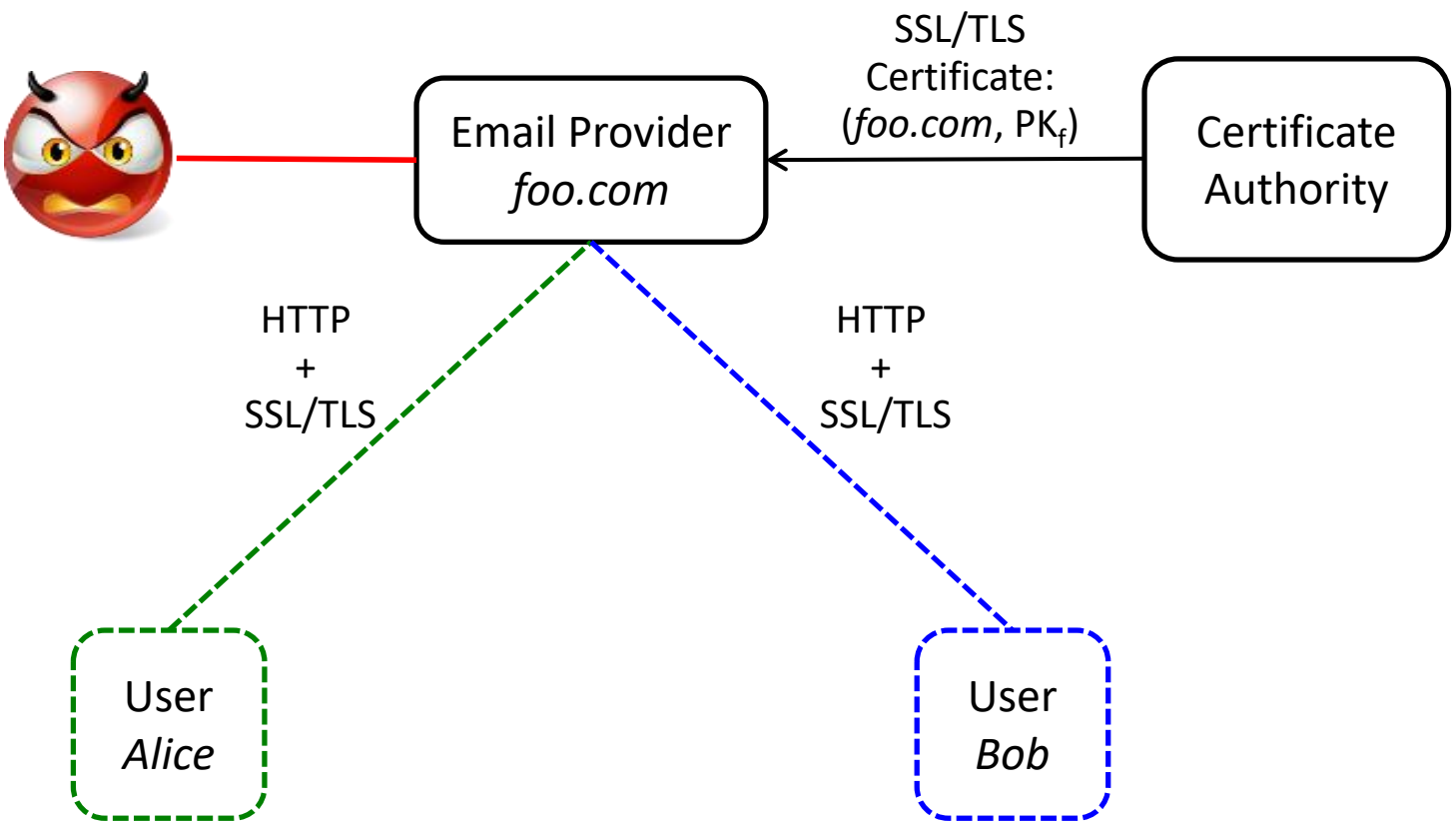
The problem of the PKI (Public key infrastructure)

- A long standing problem has been to distribute public keys securely in the presence of attackers
- Use cases:
  - Secure messaging: Alice needs to send an encrypted message to Bob and needs Bob's public key to encrypt the message
  - Web surfing and https: when Alice's browser contacts amazon.com over https, we need Amazon's PK
- Man in the middle attacker can intercept PK and return incorrect PK

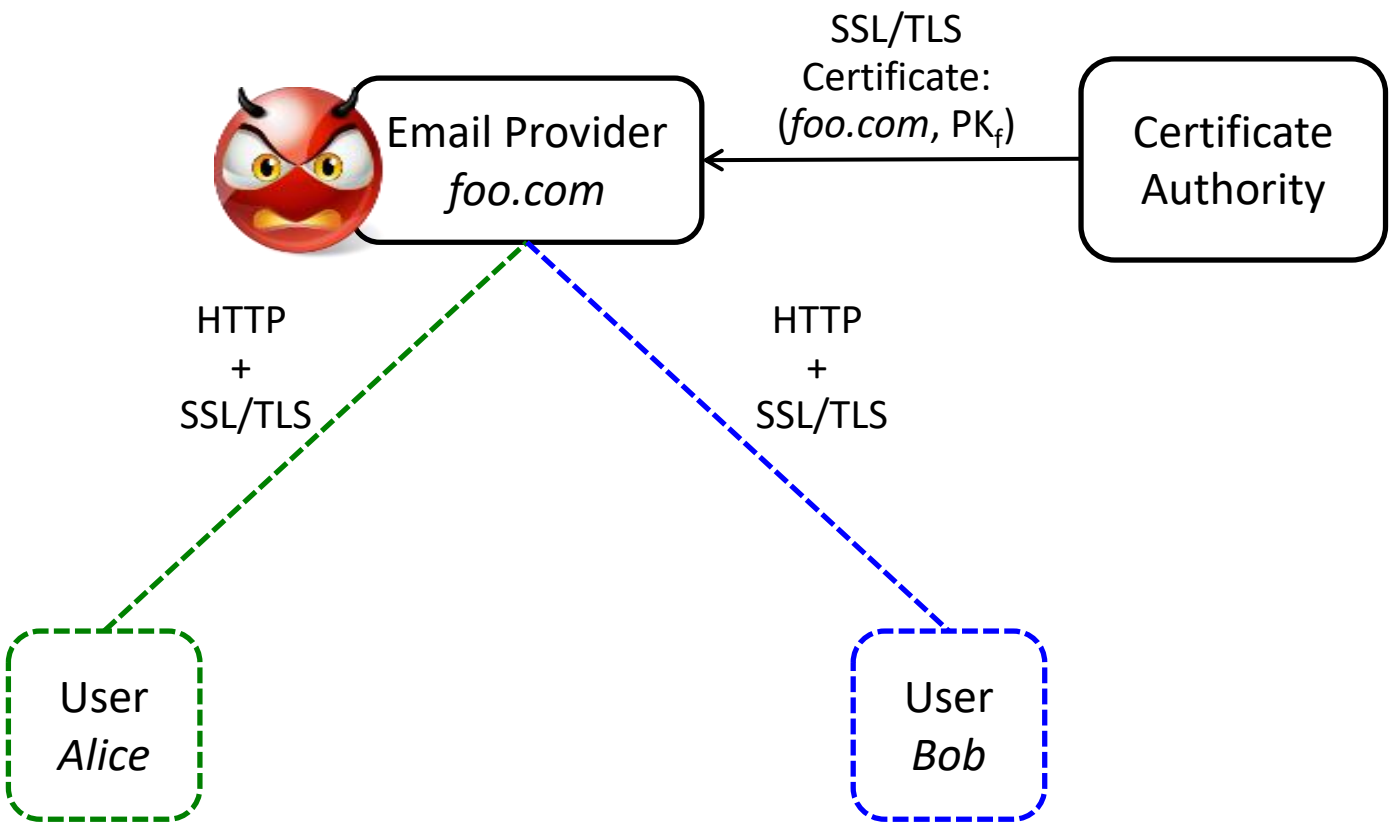
“Secure” Communication Today



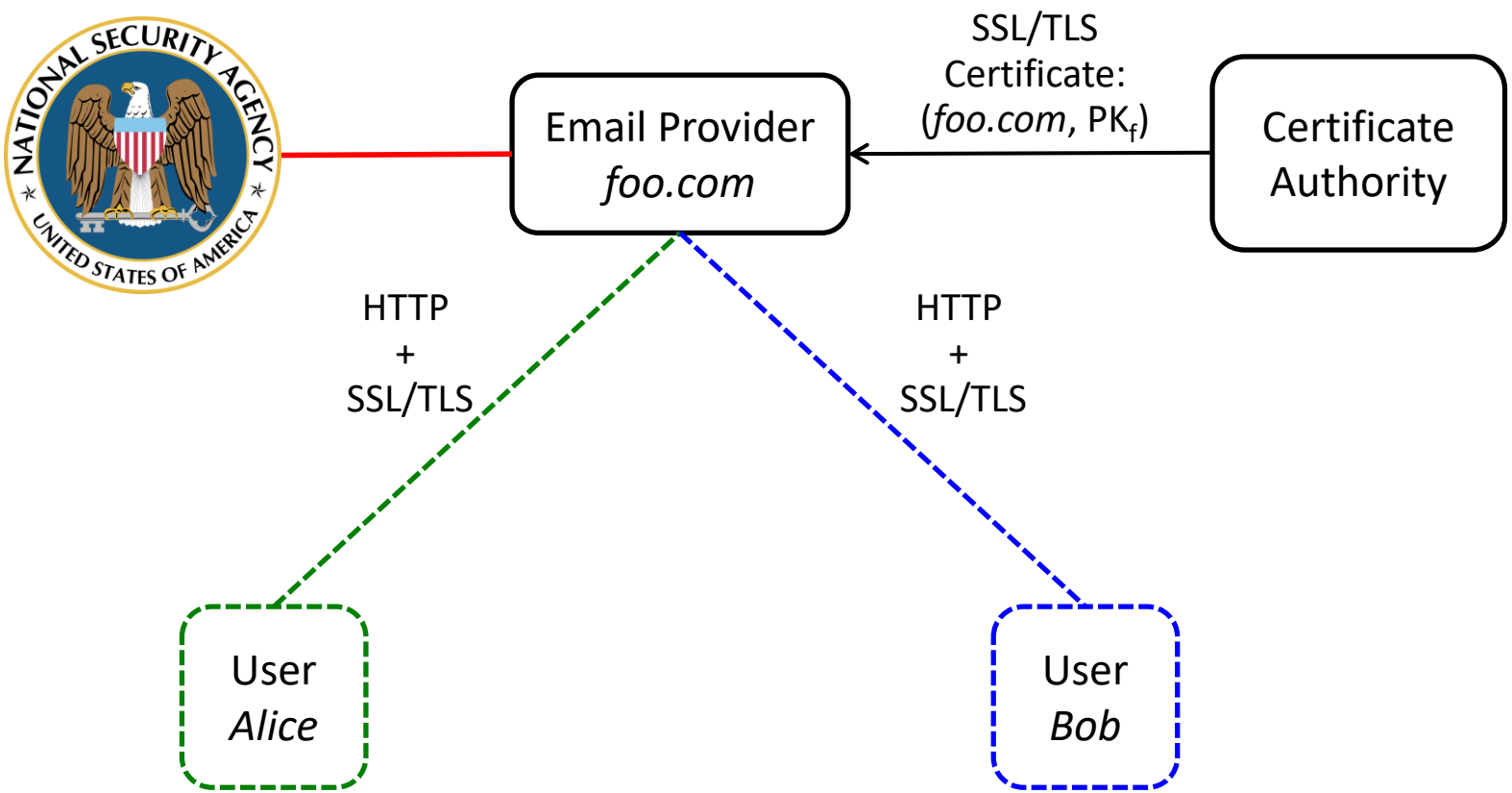
Attack: Hackers



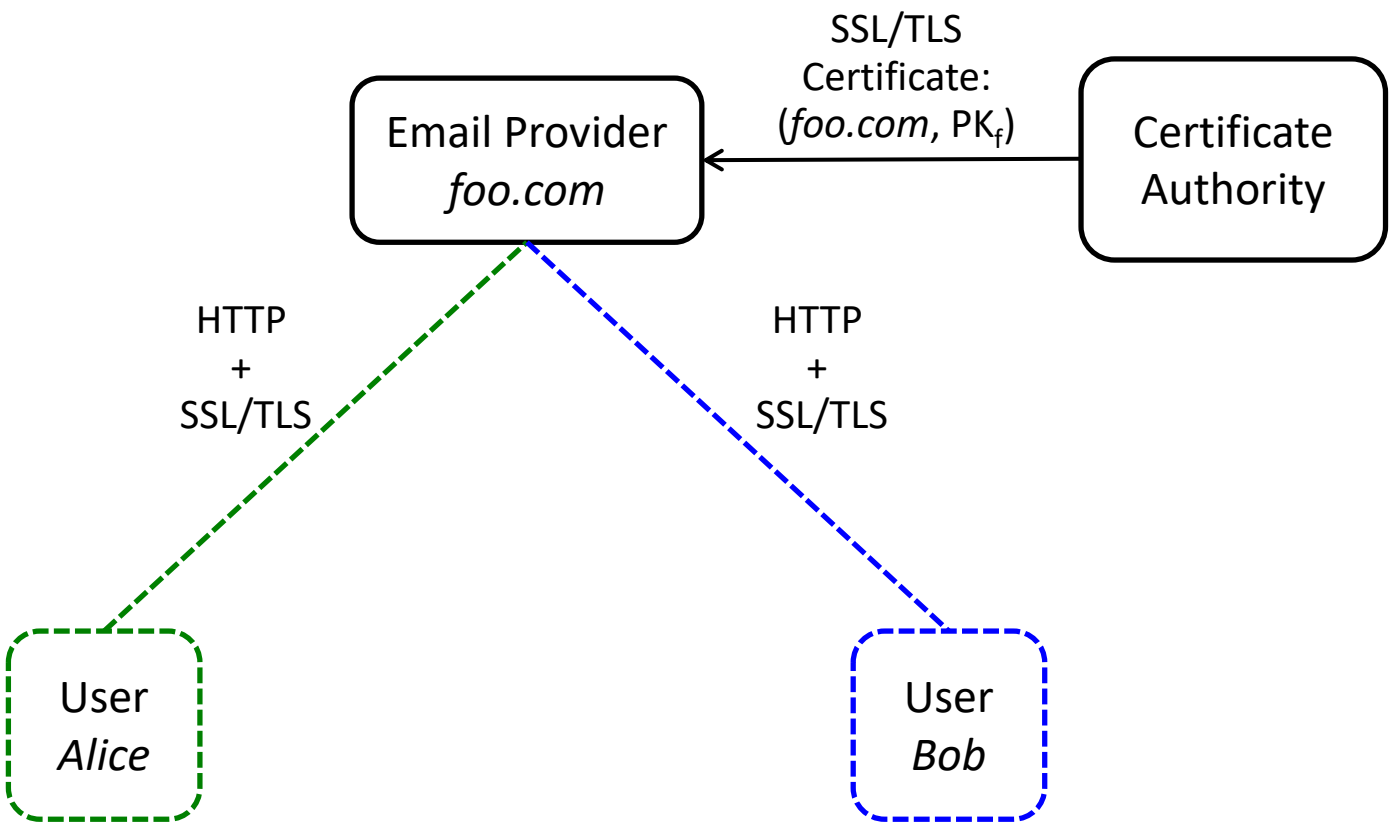
Attack: Disgruntled Employees



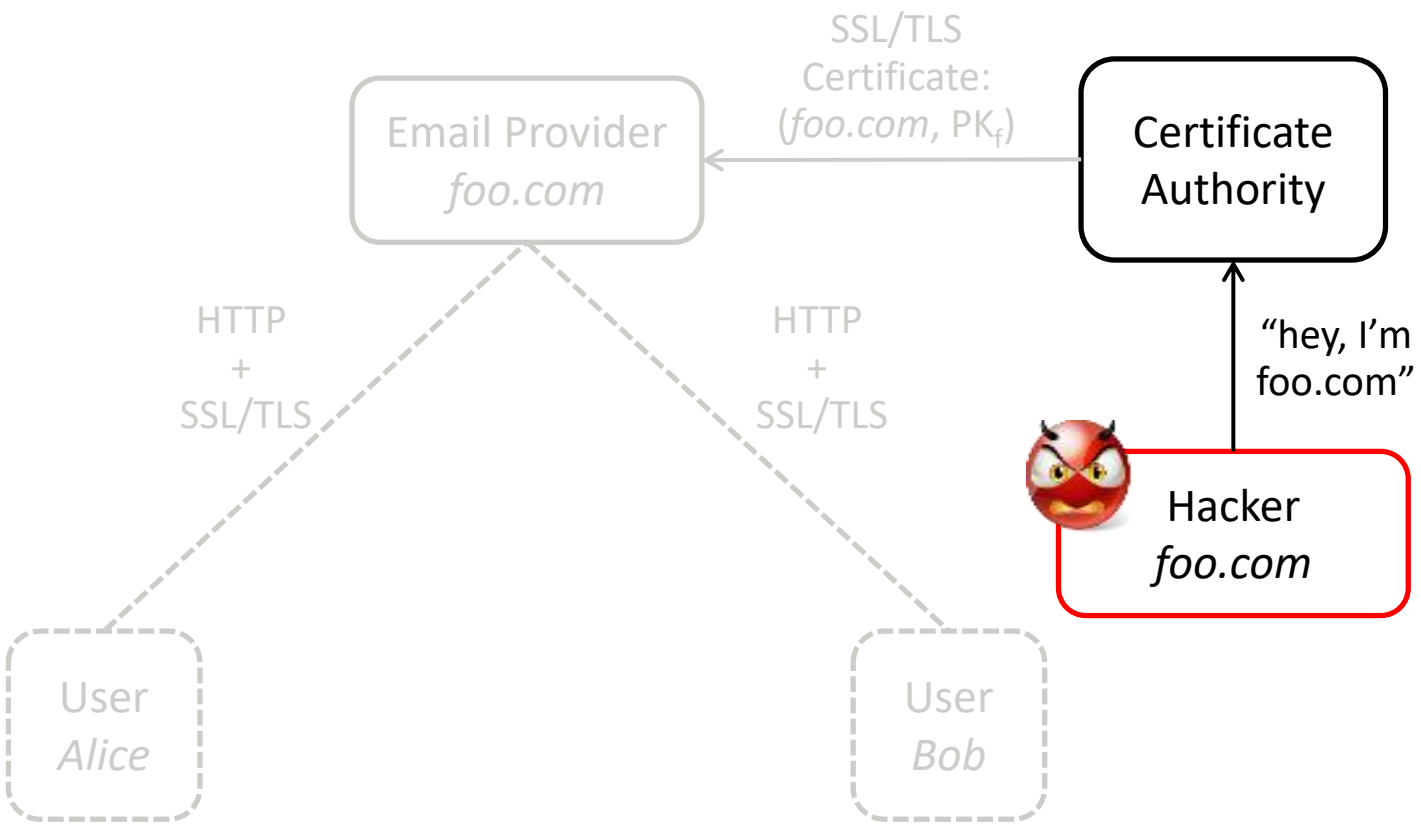
Attack: Provider Coercion



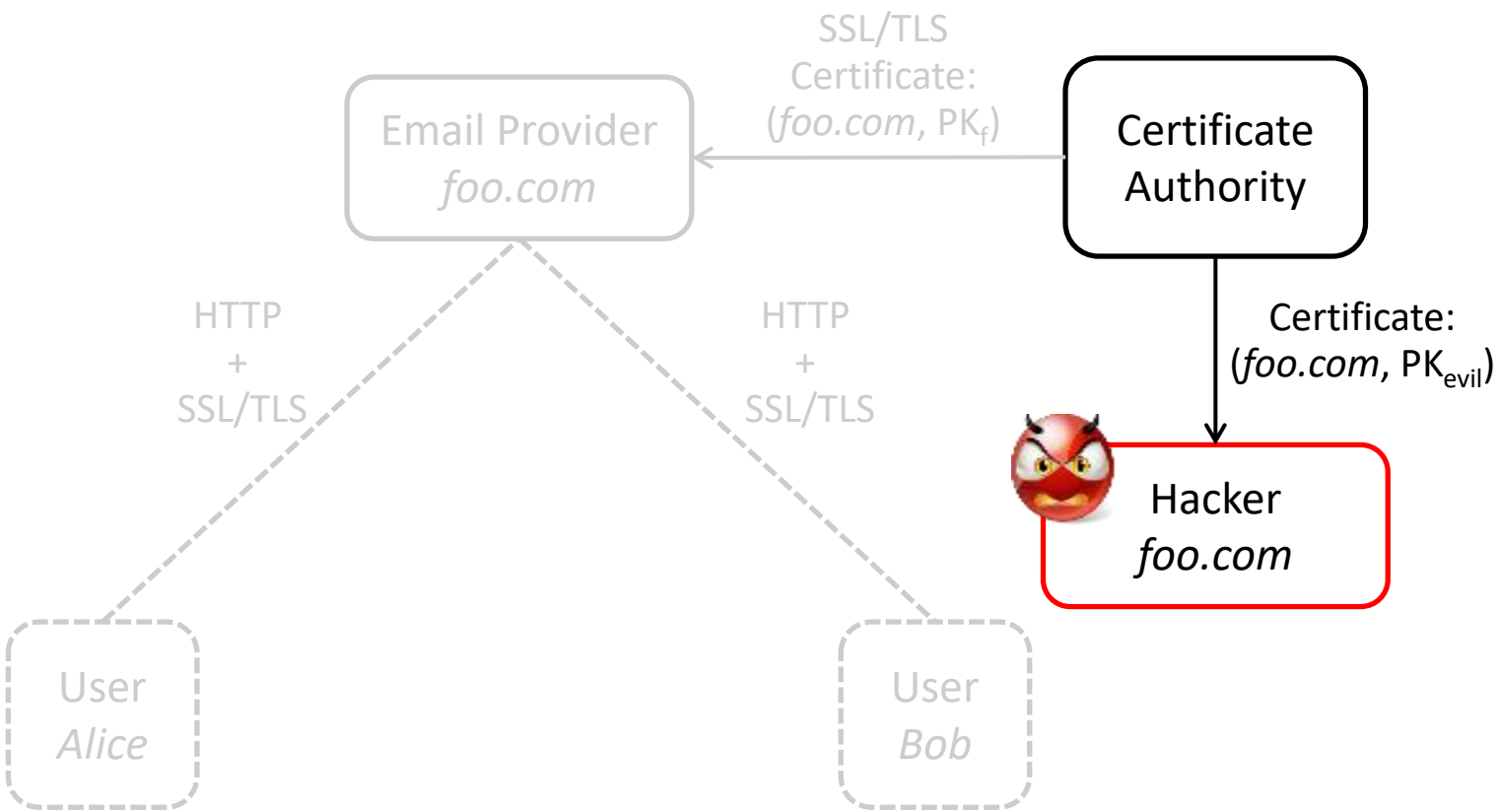
CAs are Vulnerable



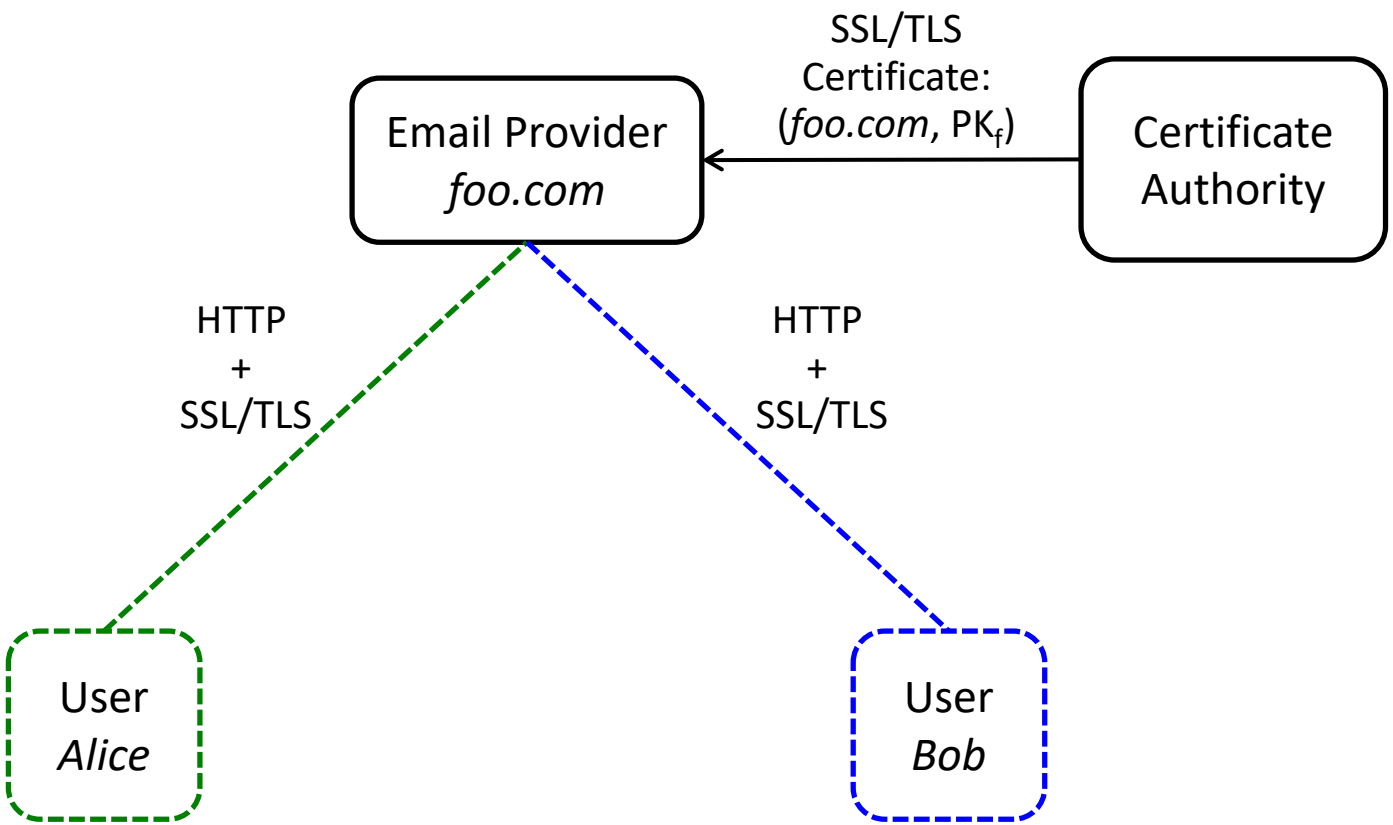
CAs are Vulnerable



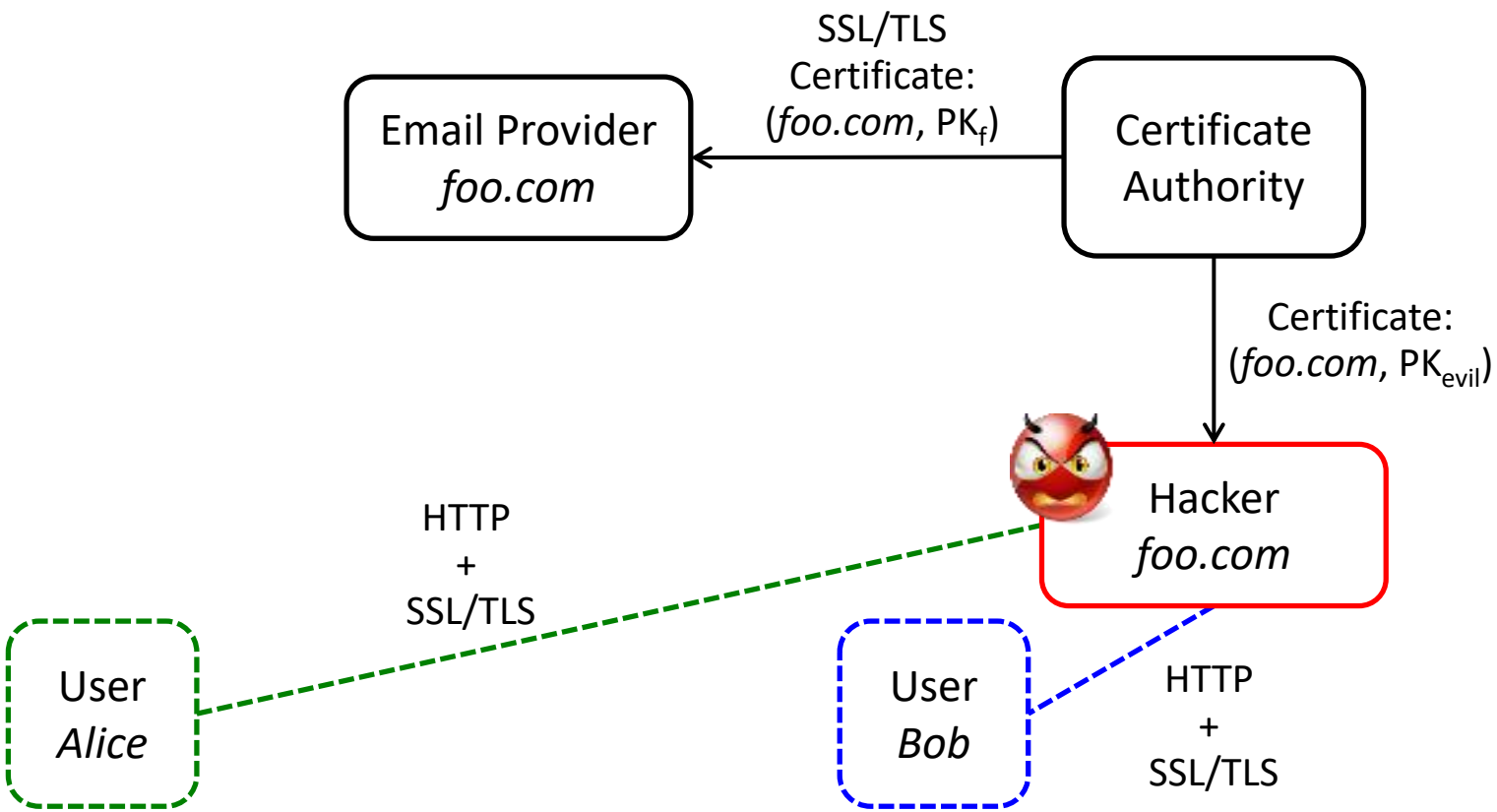
CAs are Vulnerable



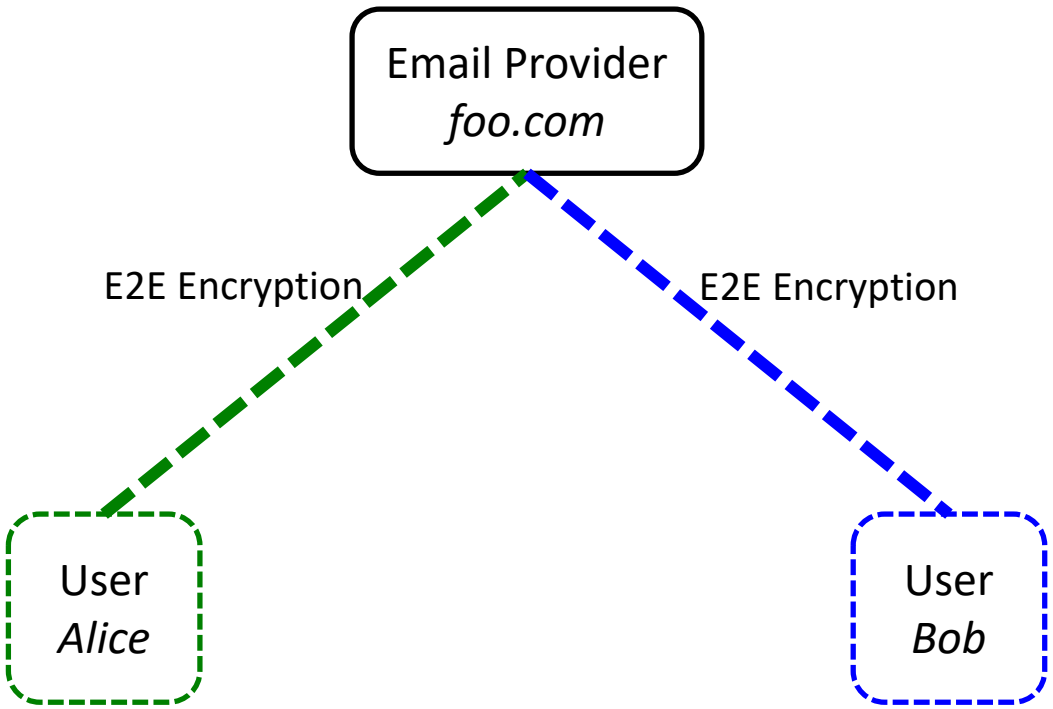
Attack: Fraudulent Certificates



Attack: Fraudulent Certificates

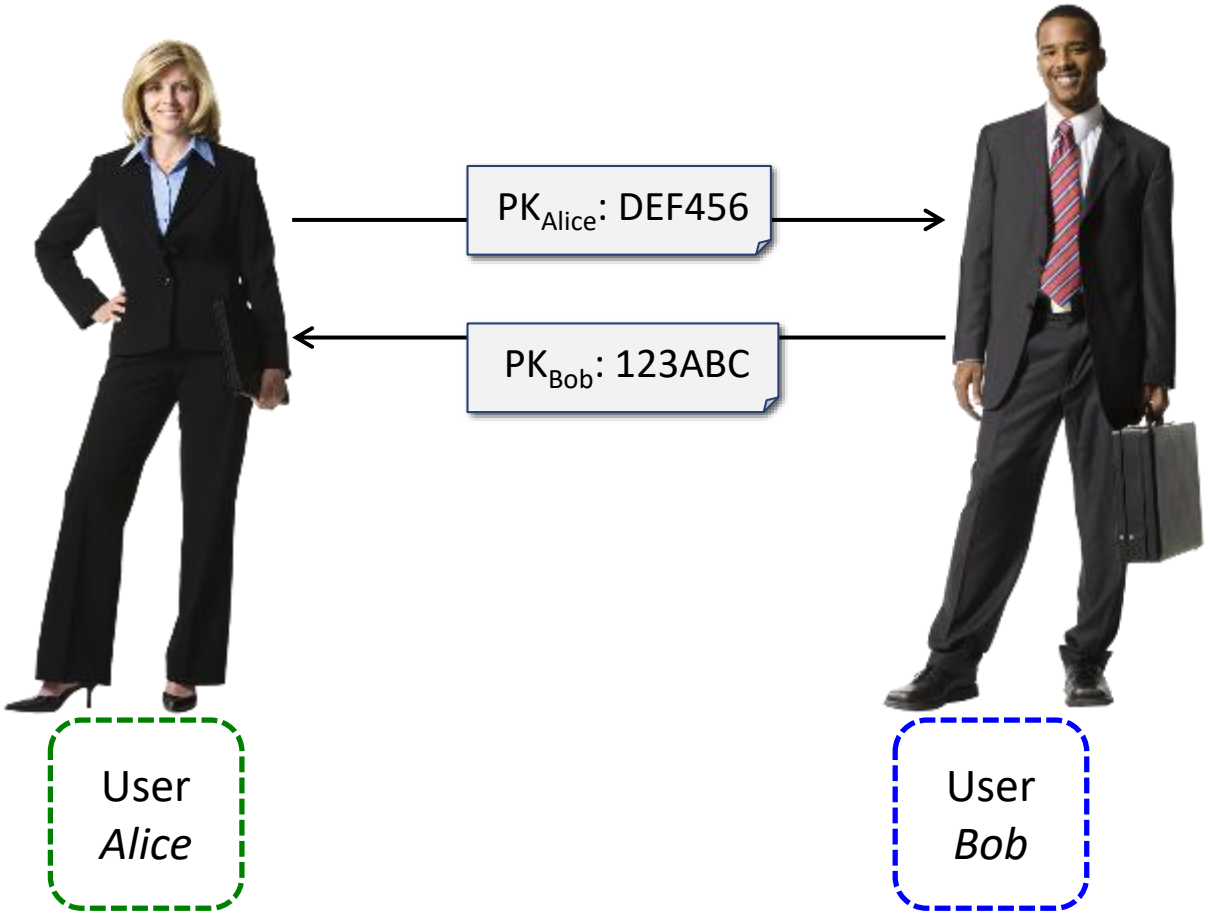


Need End-to-End Encryption…



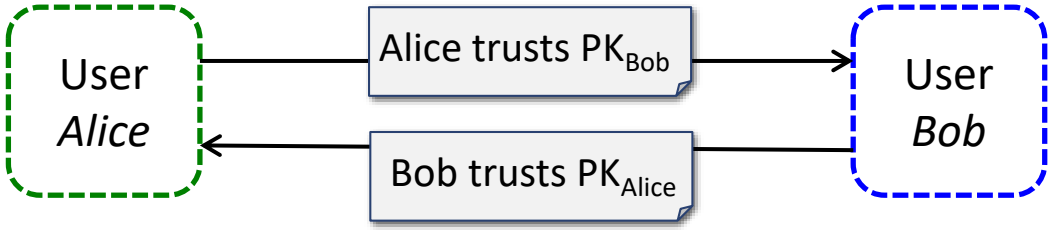
… But Key Management is Hard

Decentralized Key Management



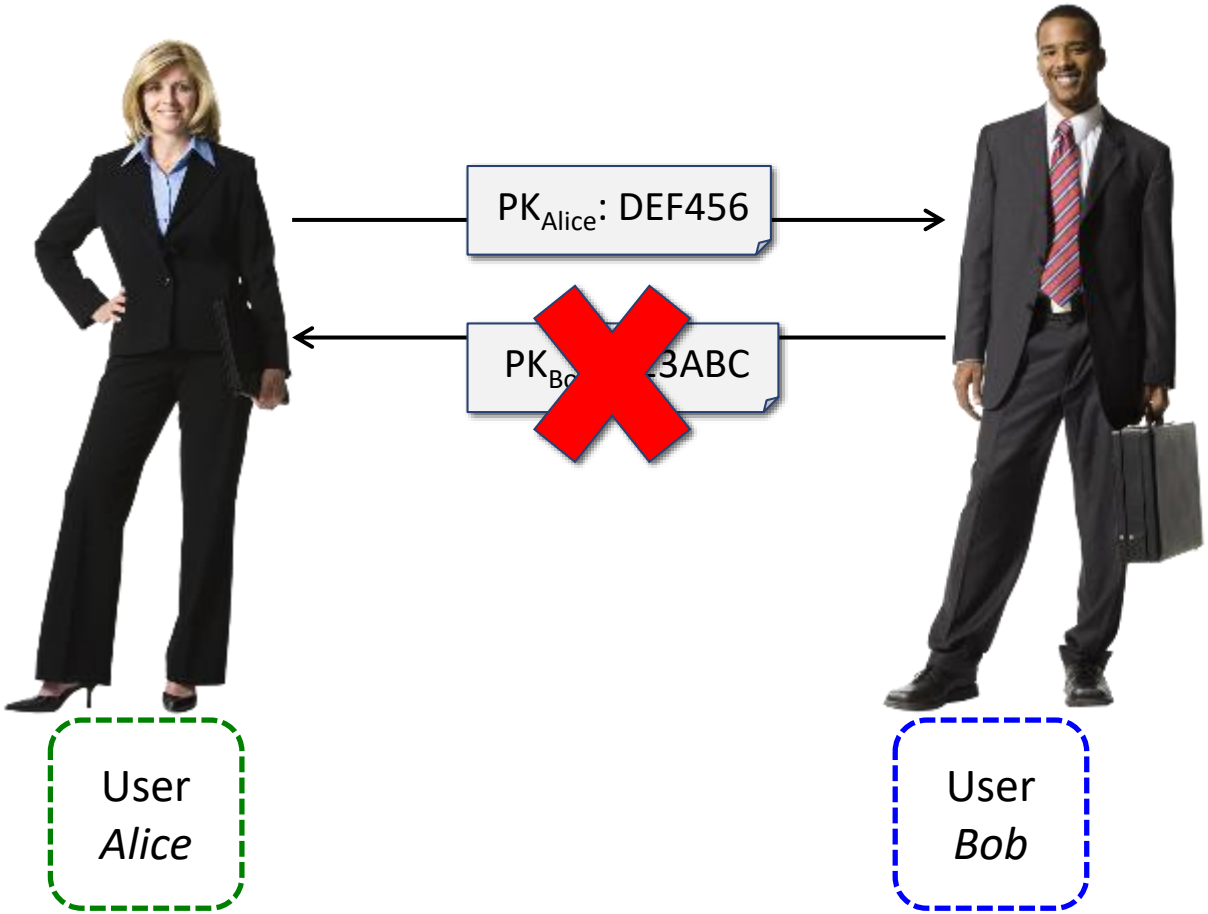
Manual key exchange

Decentralized Key Management



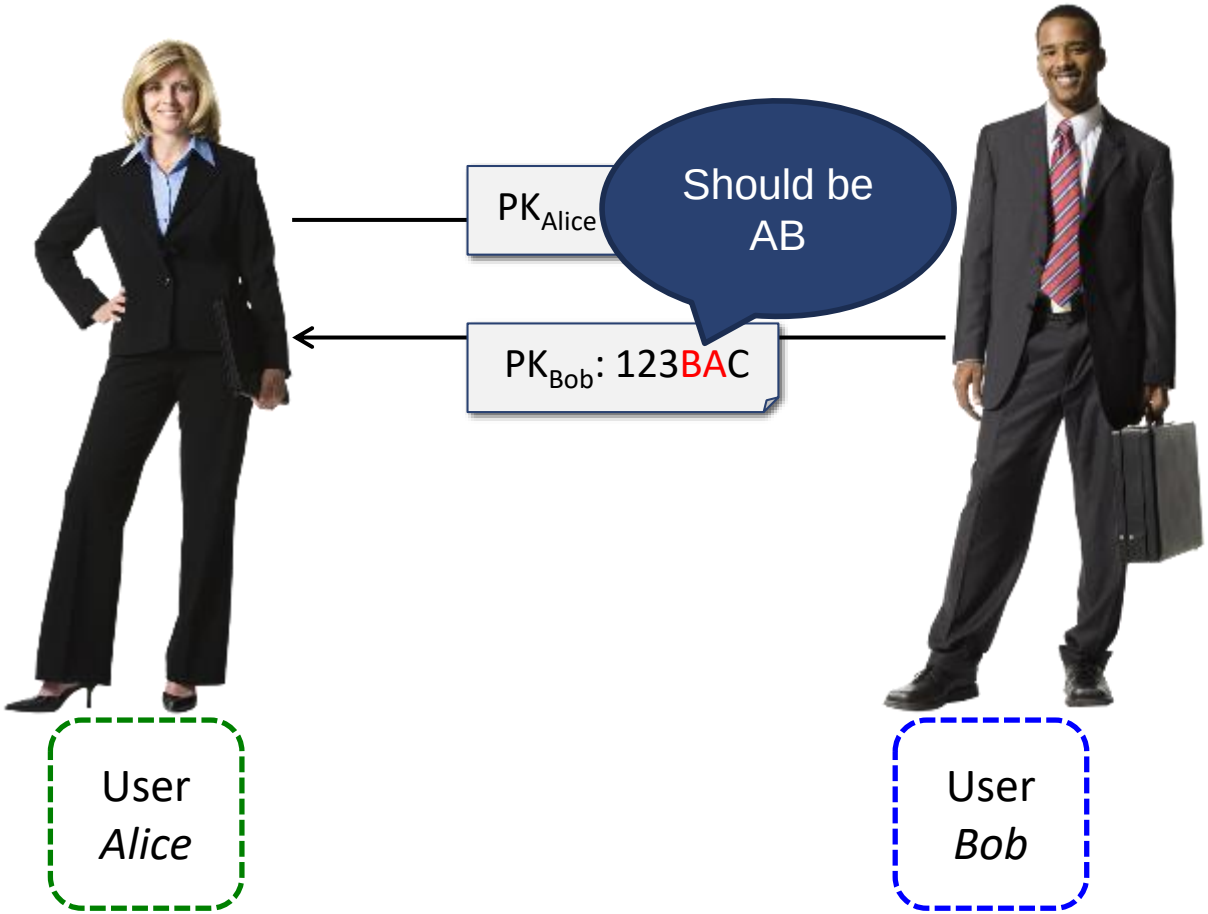
Mutual Endorsement

Decentralized KM: Pitfalls



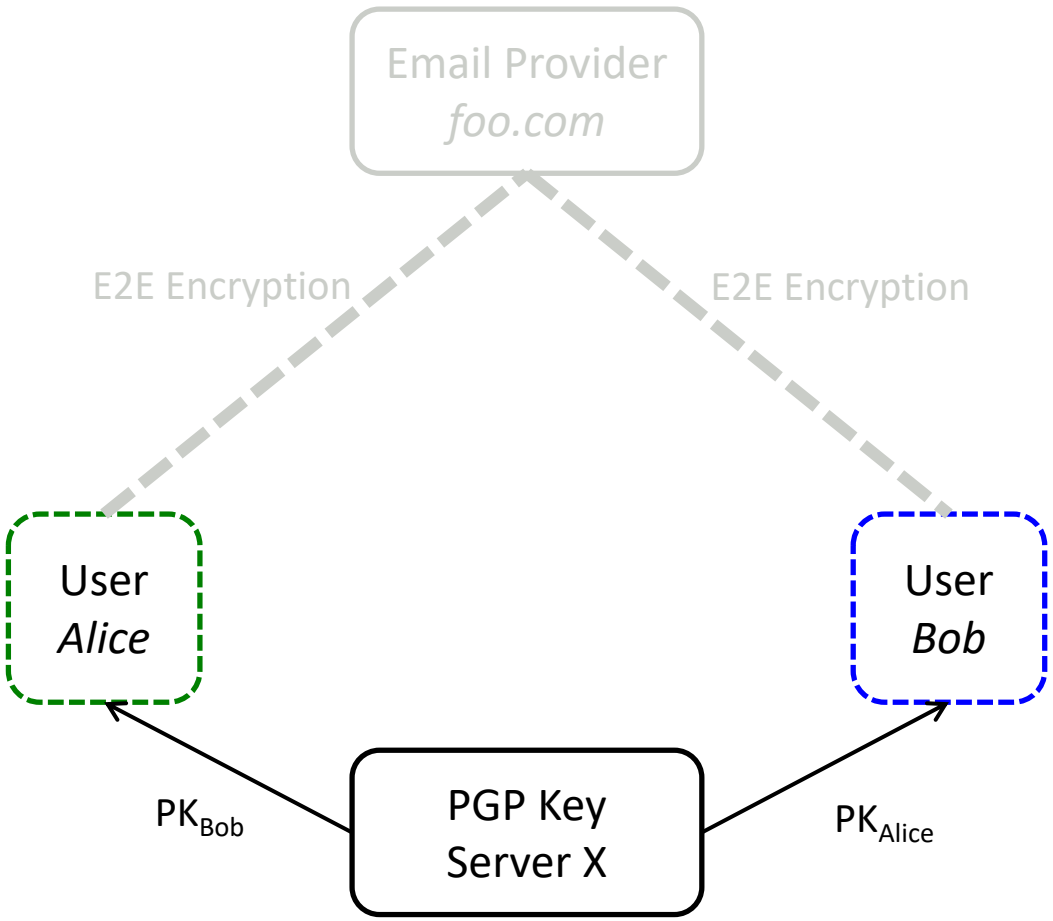
Lost keys

Decentralized KM: Pitfalls



Mistakes transferring keys

PGP Key Servers



PGP Key Servers

https://pgp.mit.edu/pks/lookup?search=marcela+melara&op=index

Search results for 'melara marcela'

Type	bits/keyID	Date	User ID
pub	4096R/99AAFE63	2016-04-08	Marcela S. Melara <msmelara@gmail.com>
pub	2048R/4AF7855E	2016-01-18	Marcela S. Melara <melara@cs.princeton.edu>
pub	4096R/CDC3499E	2015-04-11	*** KEY REVOKED *** [not verified] Marcela Melara <melara@cs.princeton.edu>
pub	2048R/69B2E80C	2012-10-01	Marcela Melara (test) <msmelara@gmail.com>

Key Signing Parties

Trusting the Web of Trust

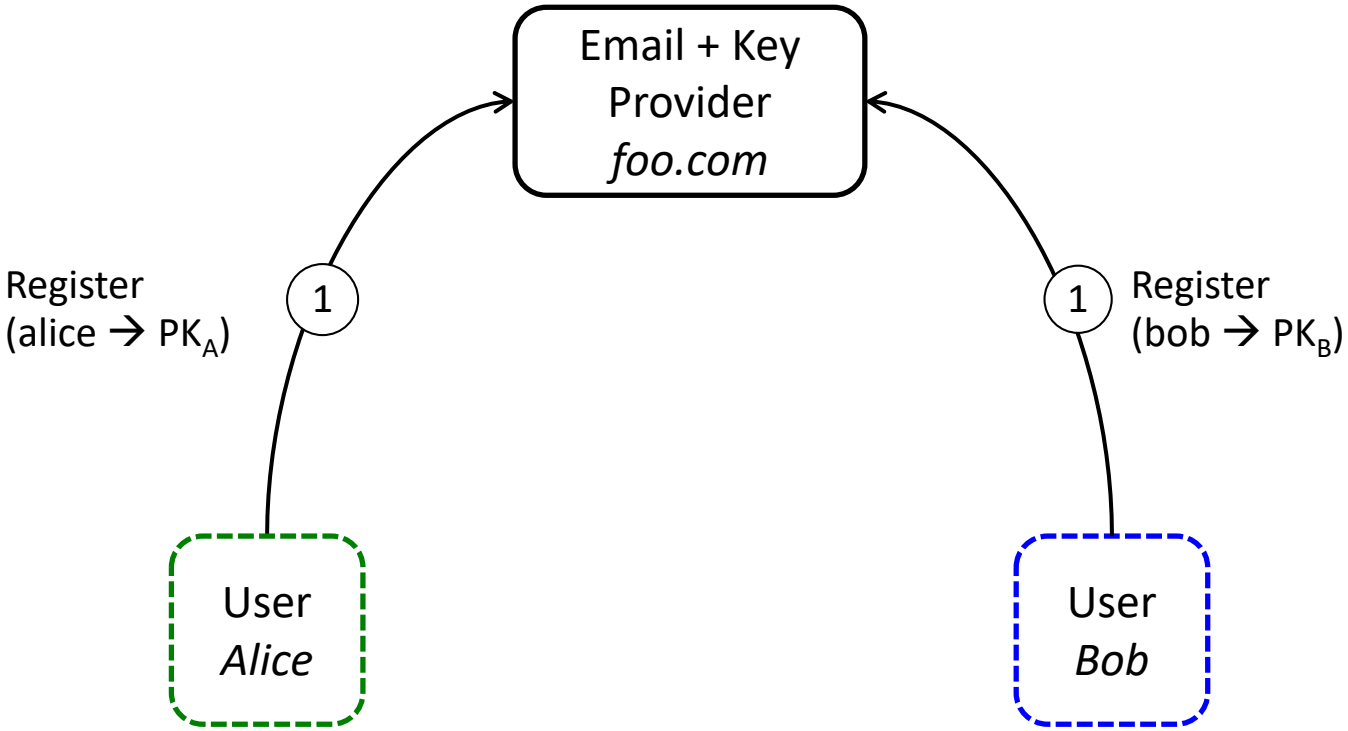
XKCD, Responsible Behavior

Crux of WoT Issues

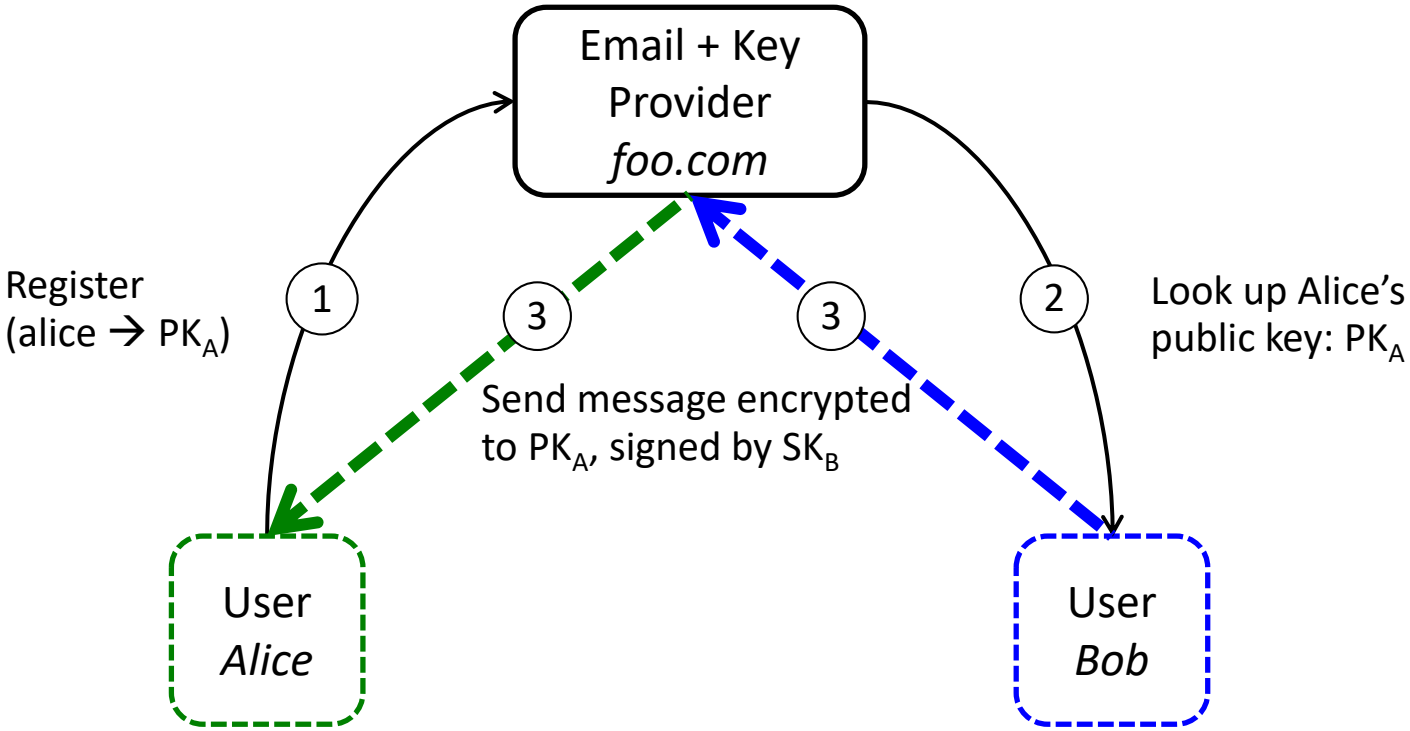
- Decentralized model: people reason about encryption.
- Studies:
  - Unintuitive and error-prone.
  - Users don’t understand encryption.
  - Leak private information.

[1] A. Whitten and J. D. Tygar. *Why Johnny can’t encrypt: a usability evaluation of PGP 5.0.* USENIX Security, Aug. 1999
[2] S. Gaw, E. W. Felten, and P. Fernandez-Kelly. *Secrecy, flagging, and paranoia: Adoption criteria in encrypted email*. CHI, Apr 2006.

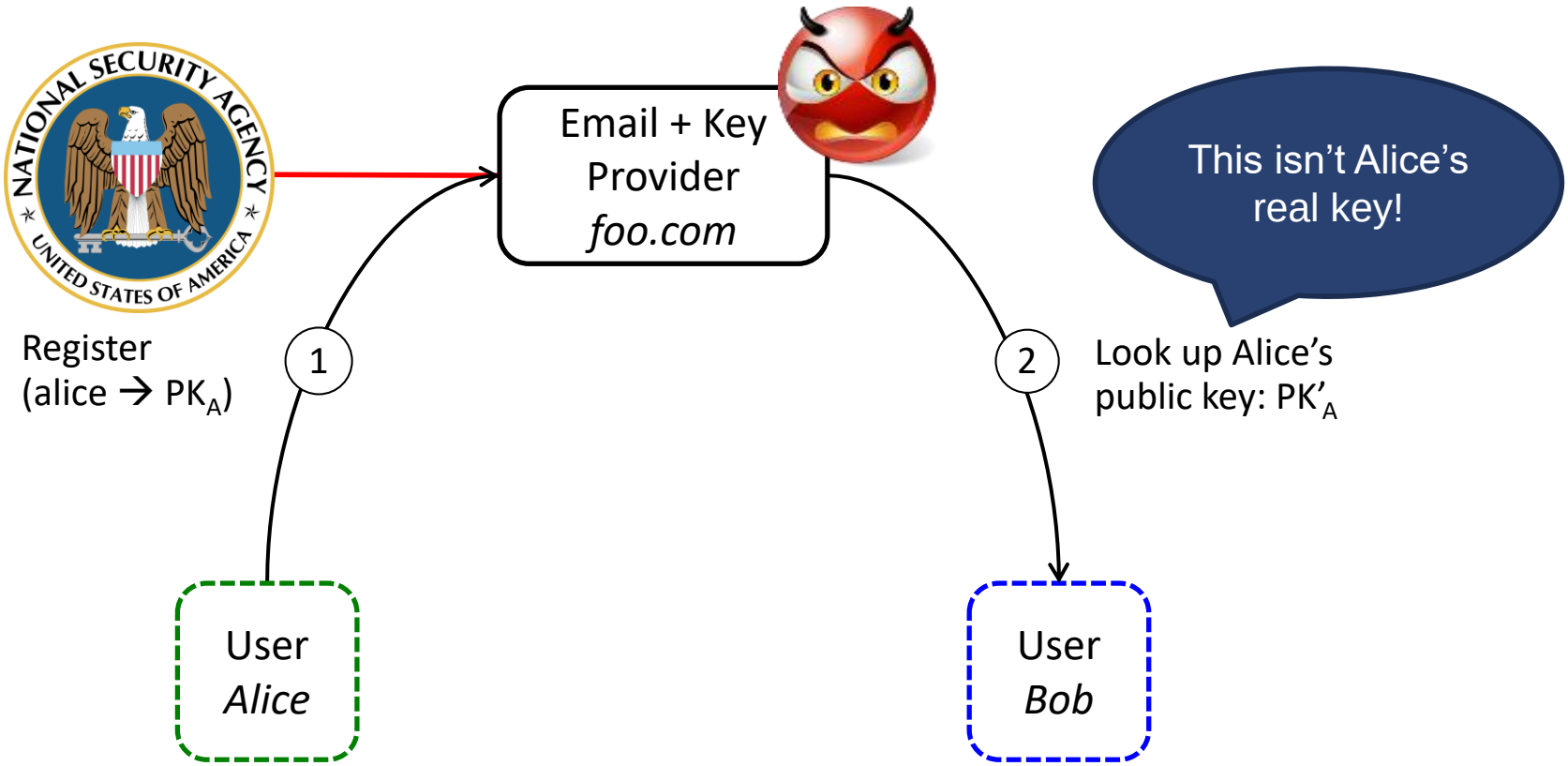
Better: Centralized Key Management



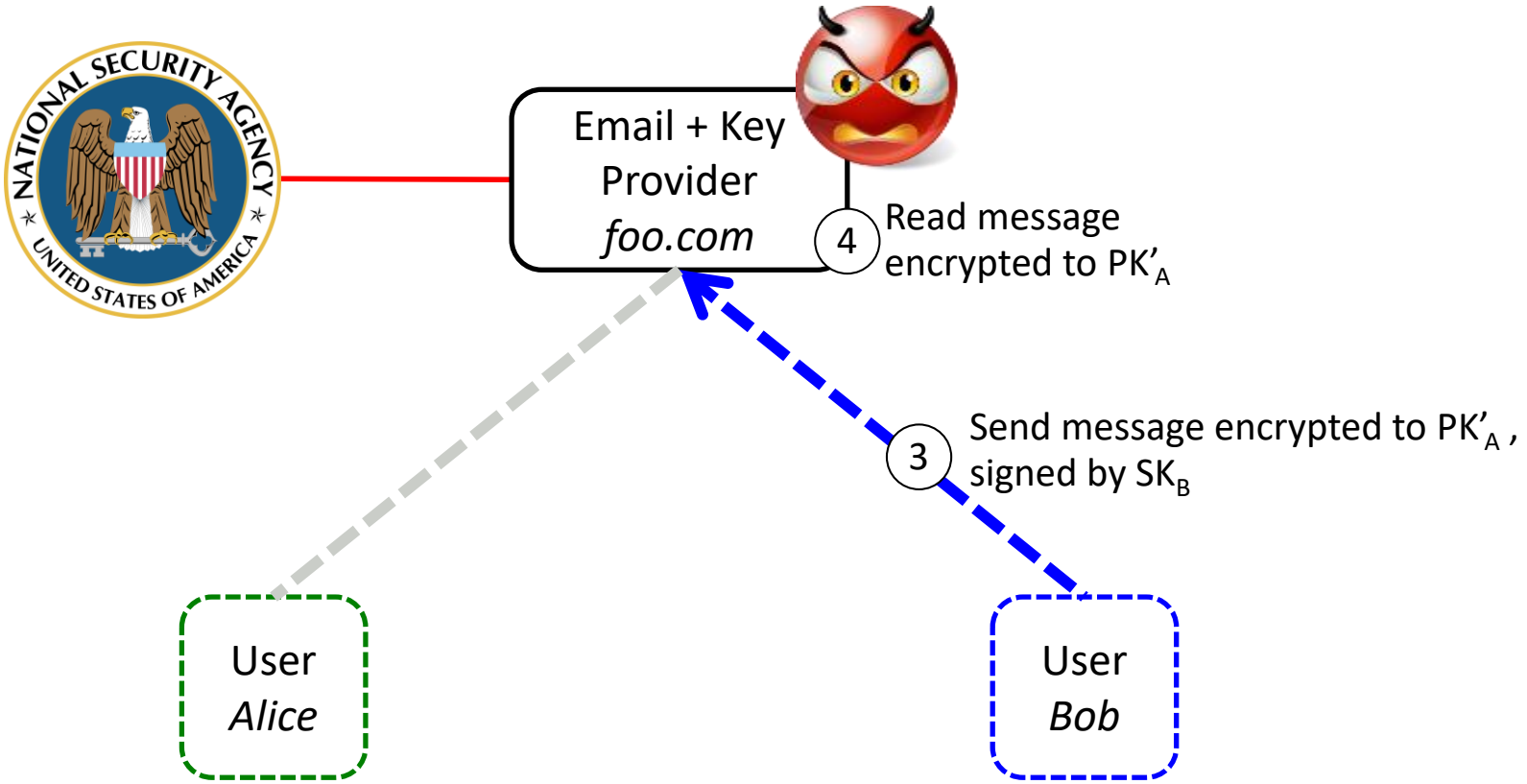
Better: Centralized Key Management



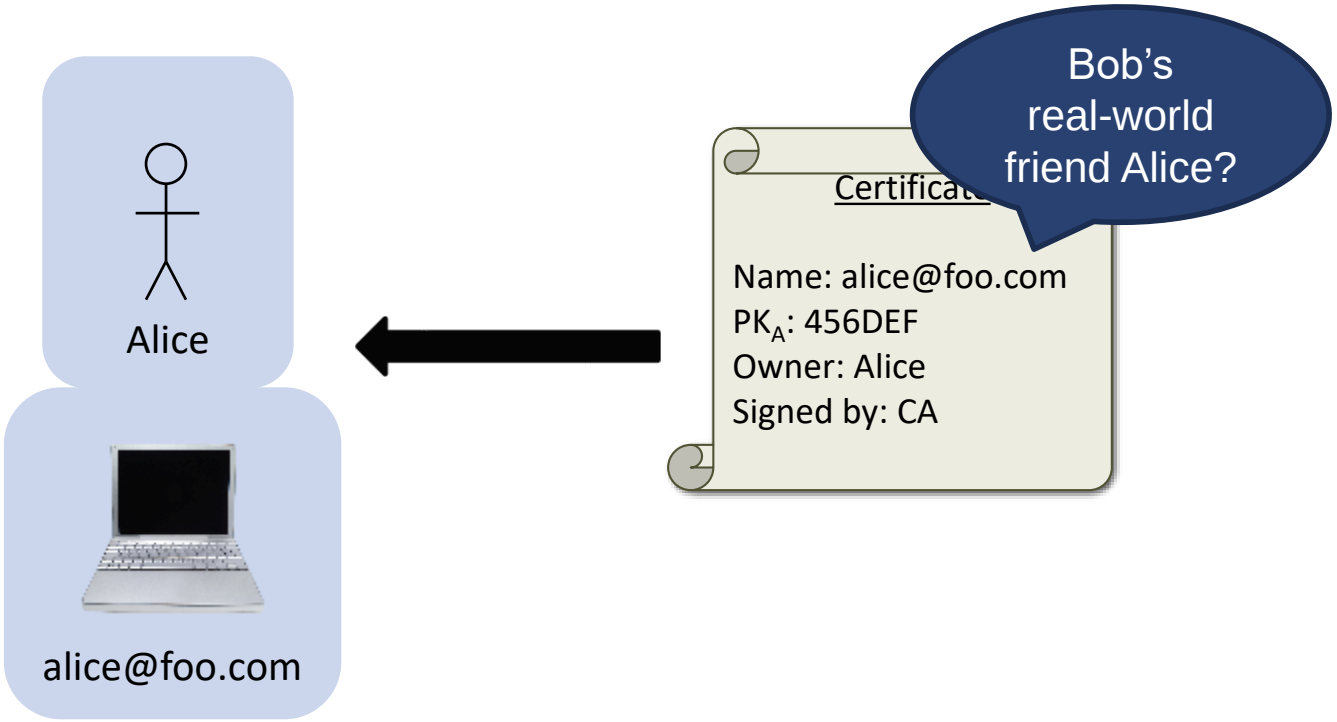
Insider Attacks, still.



Insider Attacks, still.



Old Approach: Correct Identities



New Approach: Consistent Identities

- Users expect consistency of online identities.
- Separate real-world identity from online identity.
- Certificates make no statement about correctness.

Consistency

No unexpected key changes

Alice’s key today = Alice’s key yesterday

Non-Equivocation

Key seen by Alice = Key seen by Bob

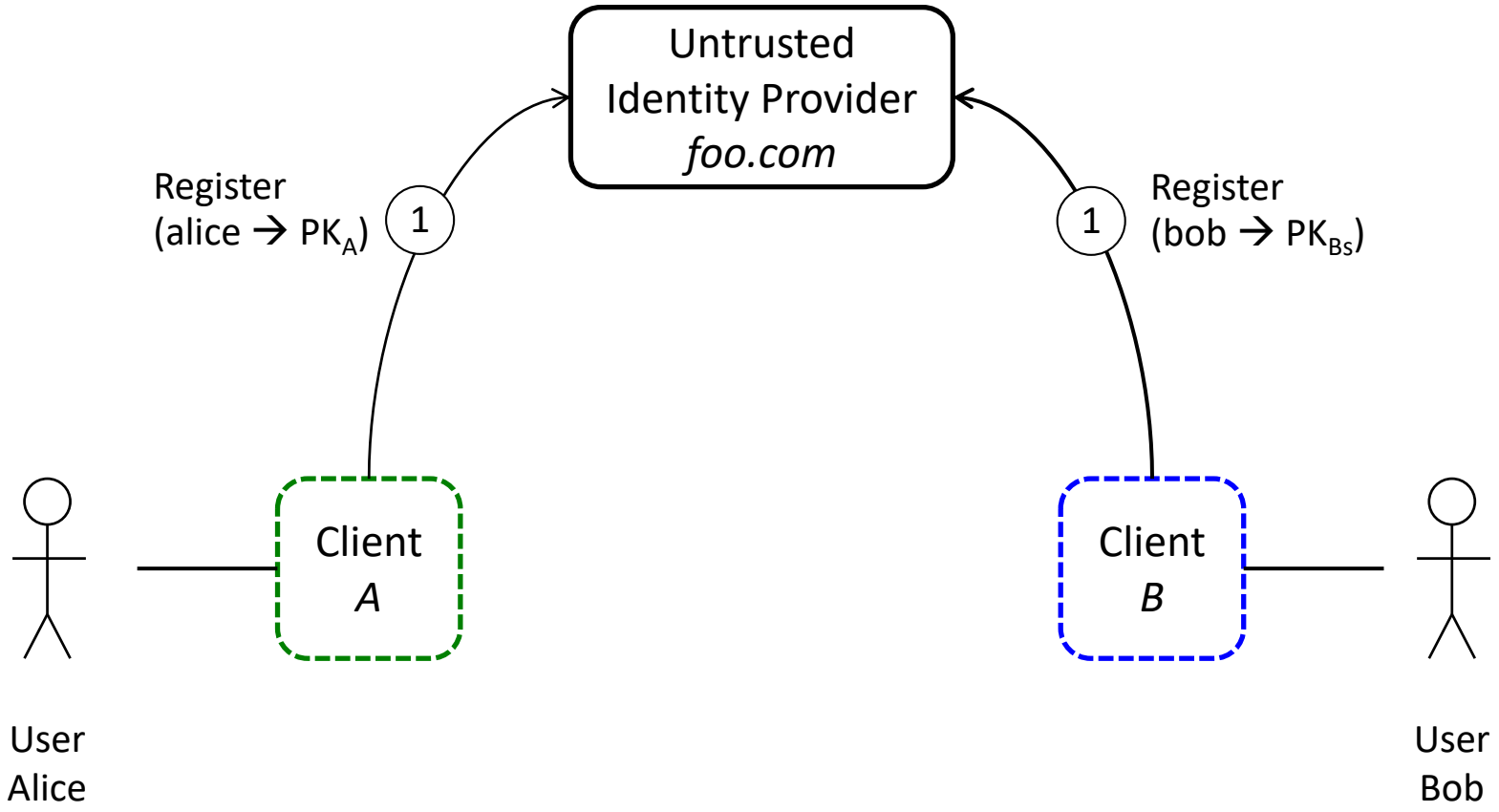
Solution: a promising new generation

- CONIKS
  - An alternative Public Key Infrastructure (PKI)
  - Embedded in Google’s project called Key Transparency or Trillian
    - The key data structure is the Log-backed Verified Map
- Certificate Transparency
  - Similar technology to Key Transparency but aimed at certificates
  - Currently mandated by Chrome, other browser support coming up

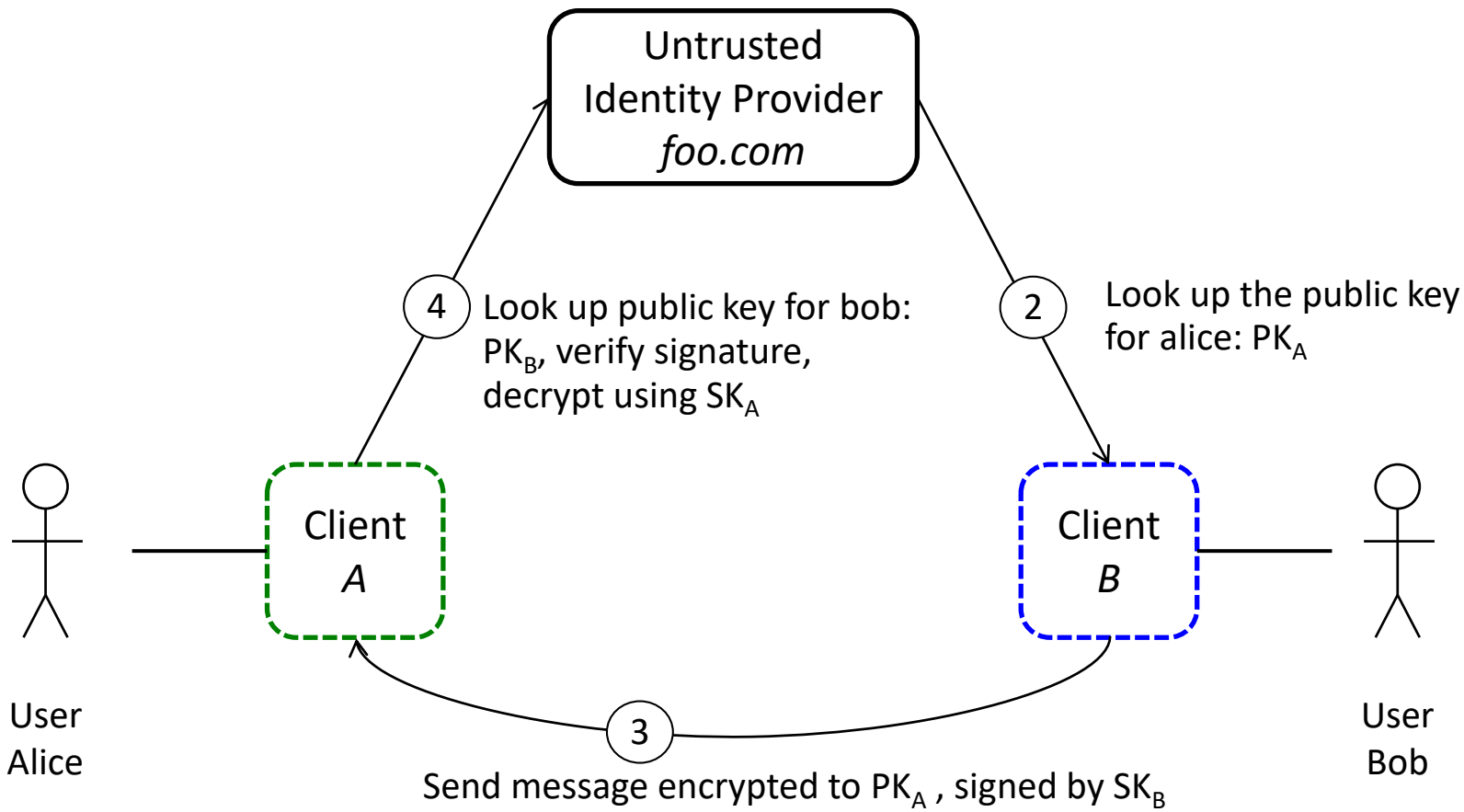
CONIKS [Melara et al. 2014]

- End-User Key Management Service.
- Consistent name-to-key bindings.
- Verifiable key directories → Untrusted identity providers.
- Clients verify consistency in-band.
  - →automated key management

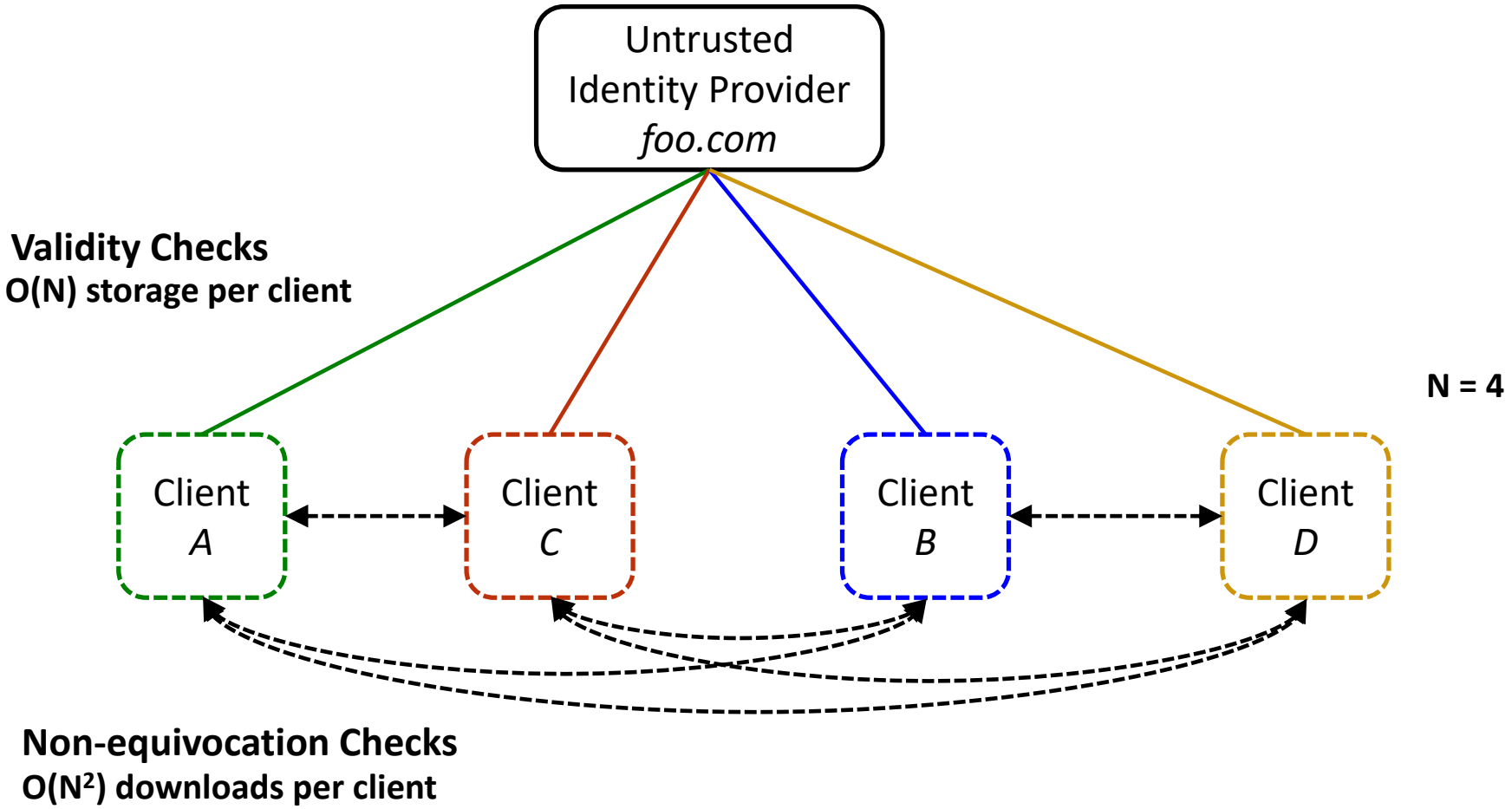
CONIKS Overview



CONIKS Overview



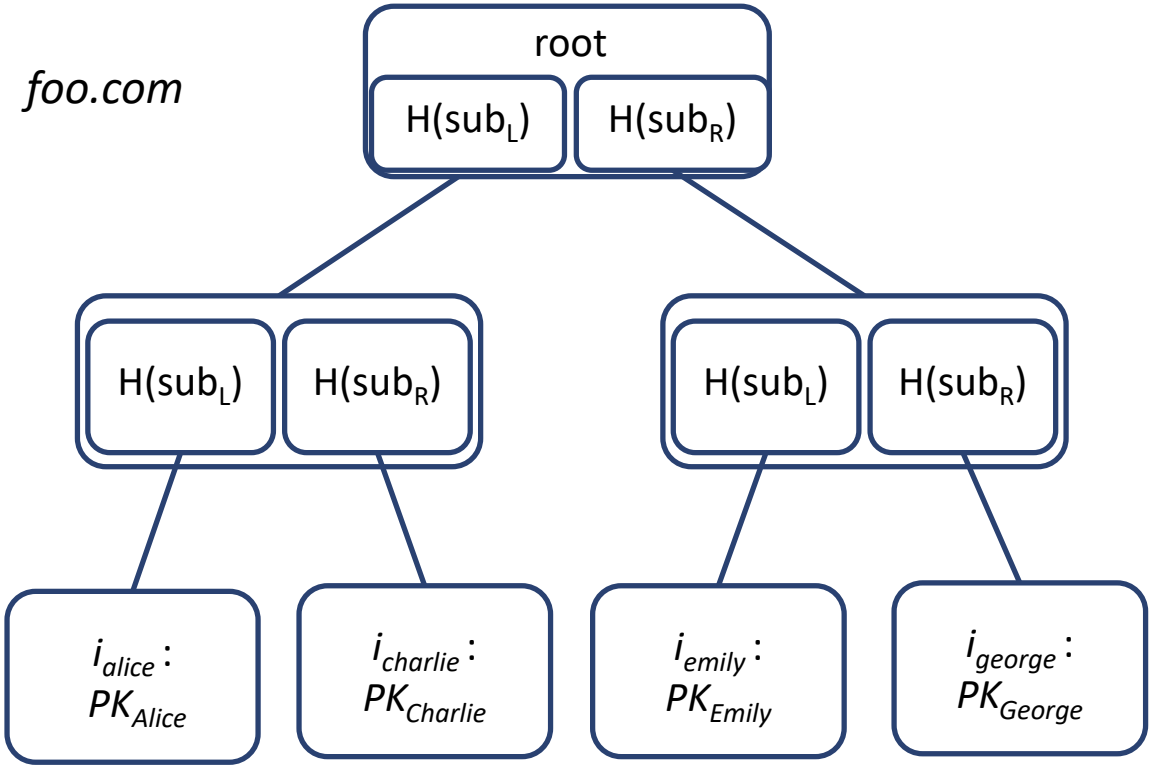
Strawman Design



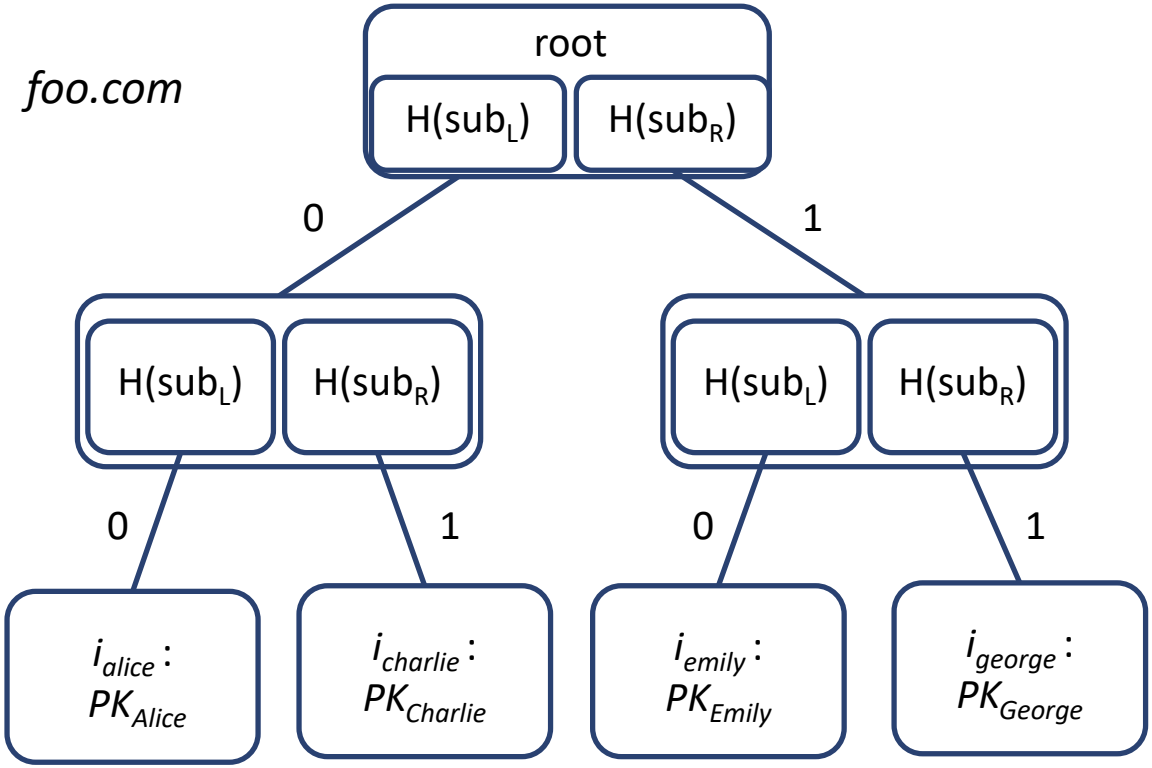
CONIKS Design

- Divide time into epochs.
- Providers generate snapshots of directory.
  → Clients do not check individual bindings.
- Providers distribute snapshots to other providers.
  → Build publicly verifiable history
- Non-repudiation: Snapshots are digitally signed.

Efficient Snapshots



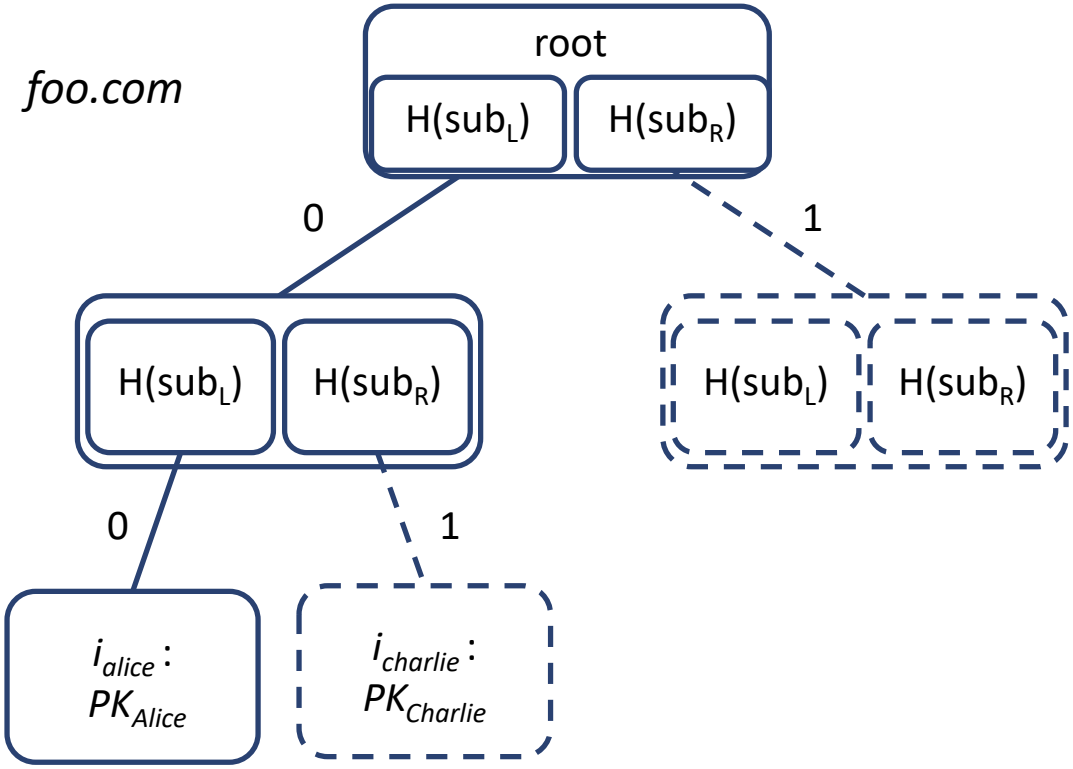
Efficient Snapshots



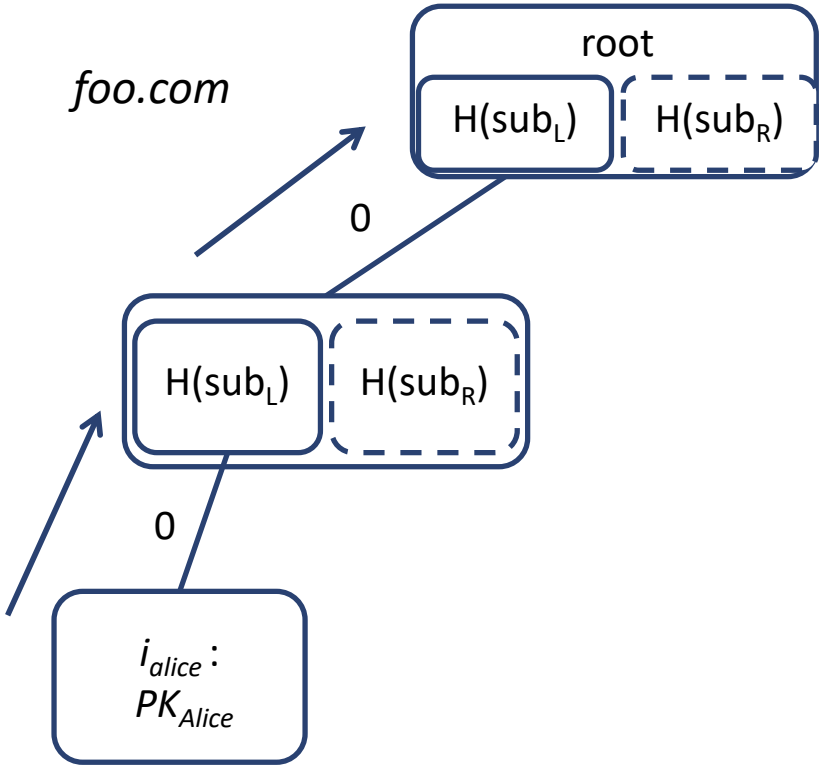
Checking Consistency

- Use snapshots to check for consistency.
- Clients need to ensure bindings are included in snapshots.
  - Lookups: prove inclusion of bindings in directory.
- Clients check validity, non-equivocation.
- Providers cross-verify each other for non-equivocation.

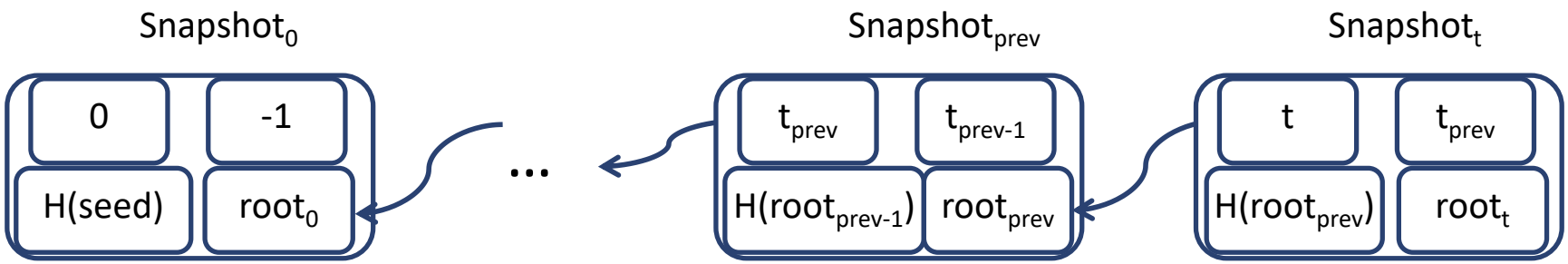
Proving Inclusion: Authentication Paths



Proving Inclusion: Authentication Paths

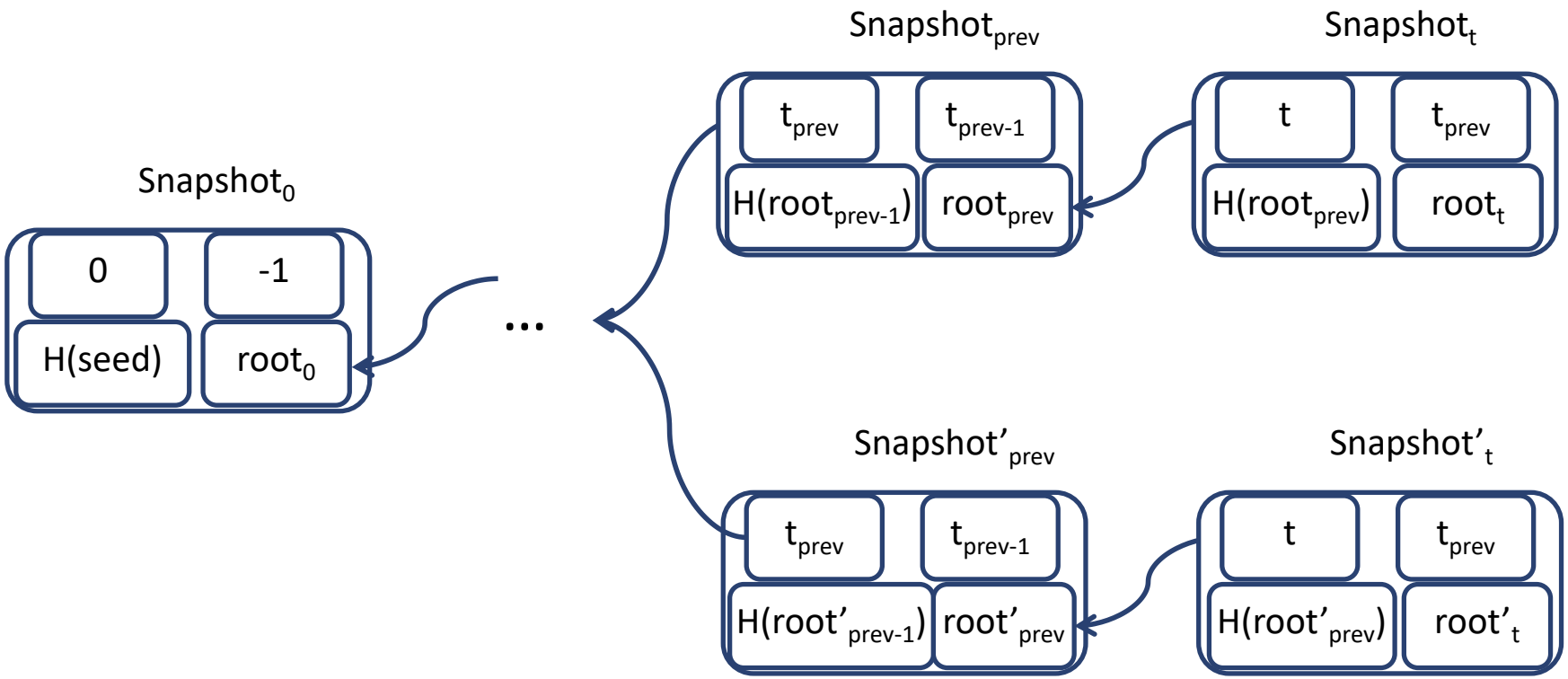


Checking Non-Equivocation: Snapshot History



Trillian calls this: Log-backed Verified Map

Checking Non-Equivocation: Snapshot History



The server can try a fork attack, but after fork the provider must maintain these forked hash chains for the rest of time, and not allow clients seeing one branch of the hash chain to communicate with anyone seeing the other branch.

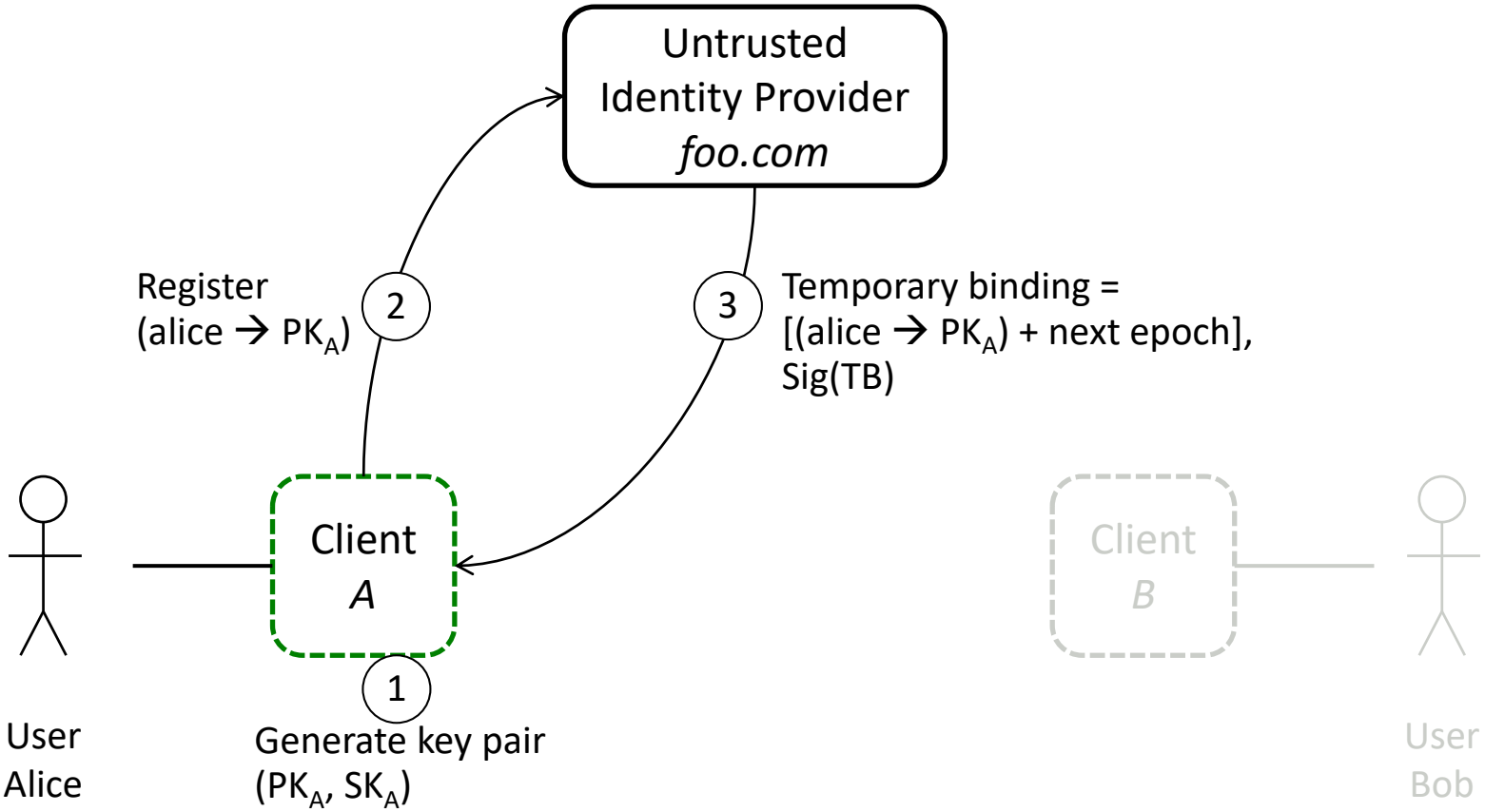
CONIKS Protocols

- Registration: New bindings/updated keys.
- Lookups: Clients check bindings are included in directory.
- Monitoring: Clients check for validity of bindings.
- Auditing: Clients & providers check for non-equivocation.

CONIKS Protocols

- Registration: New bindings/updated keys.
- Lookups: Clients check bindings are included in directory.
- Monitoring: Clients check for validity of bindings.
- Auditing: Clients & providers check for non-equivocation.

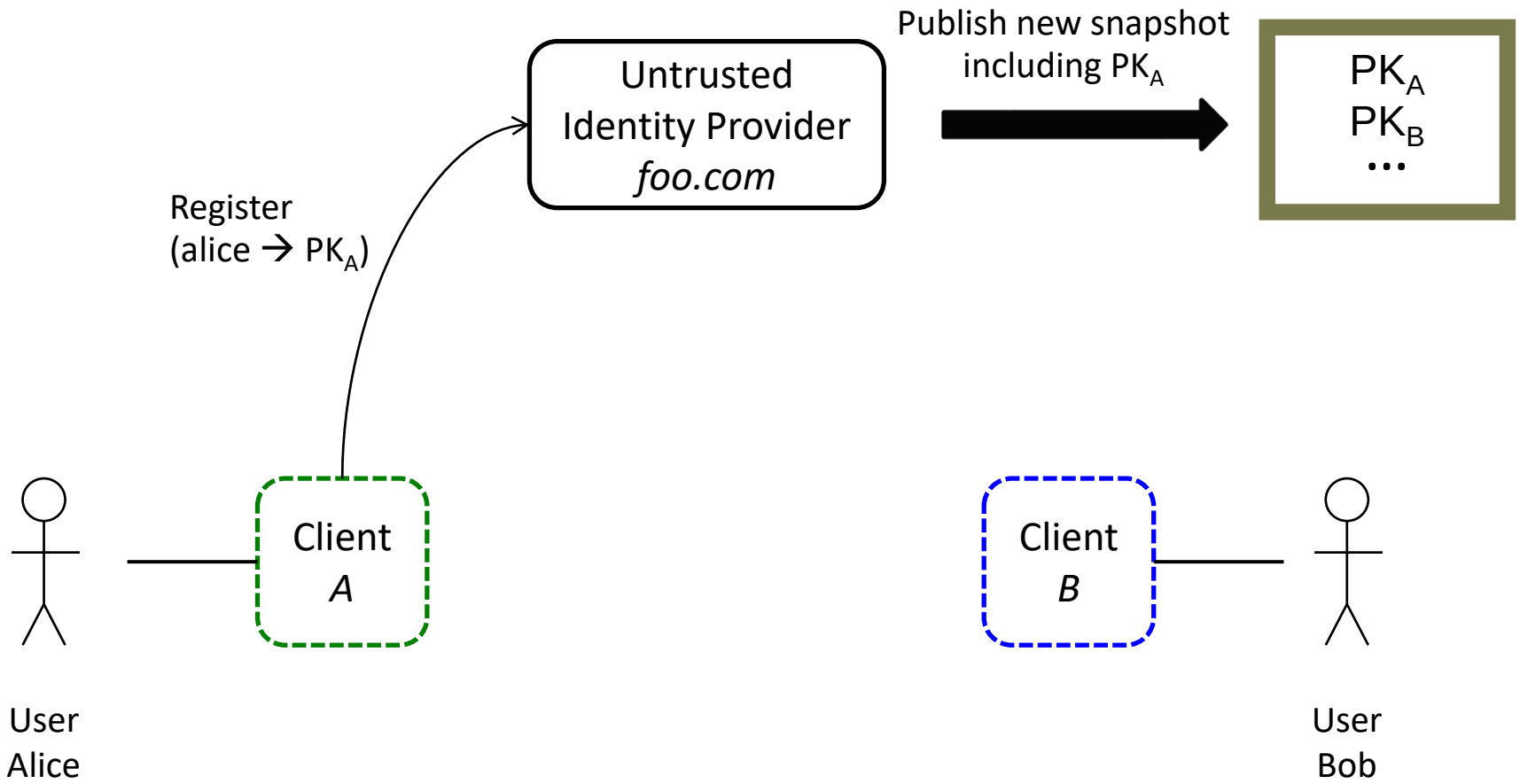
Registration Protocol



CONIKS Protocols

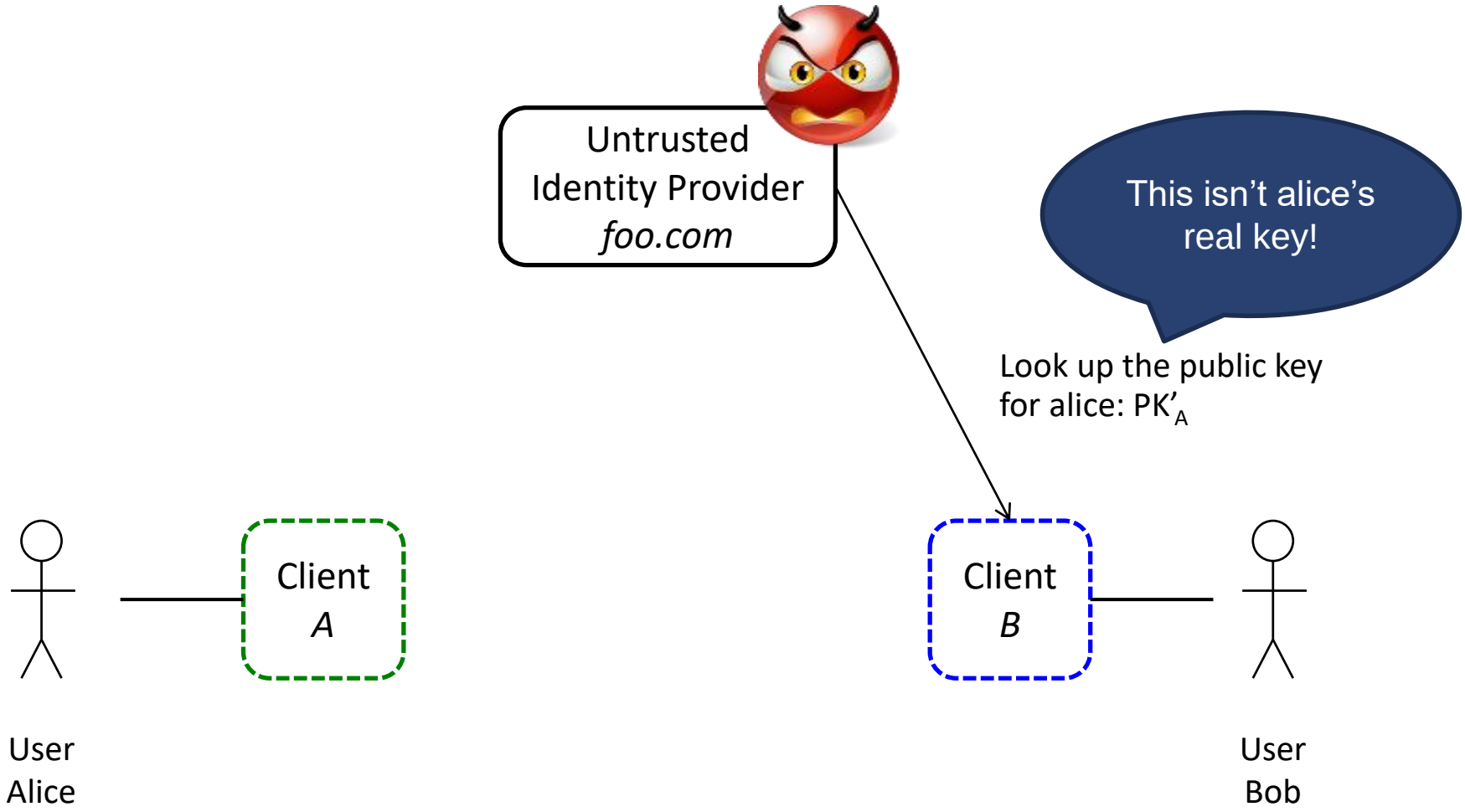
- Registration: new bindings/updated keys.
- Lookups: Clients check bindings are included in directory.
- Monitoring: Clients check for validity of bindings.
- Auditing: Clients & providers check for non-equivocation.

Lookups without Inclusion Proofs



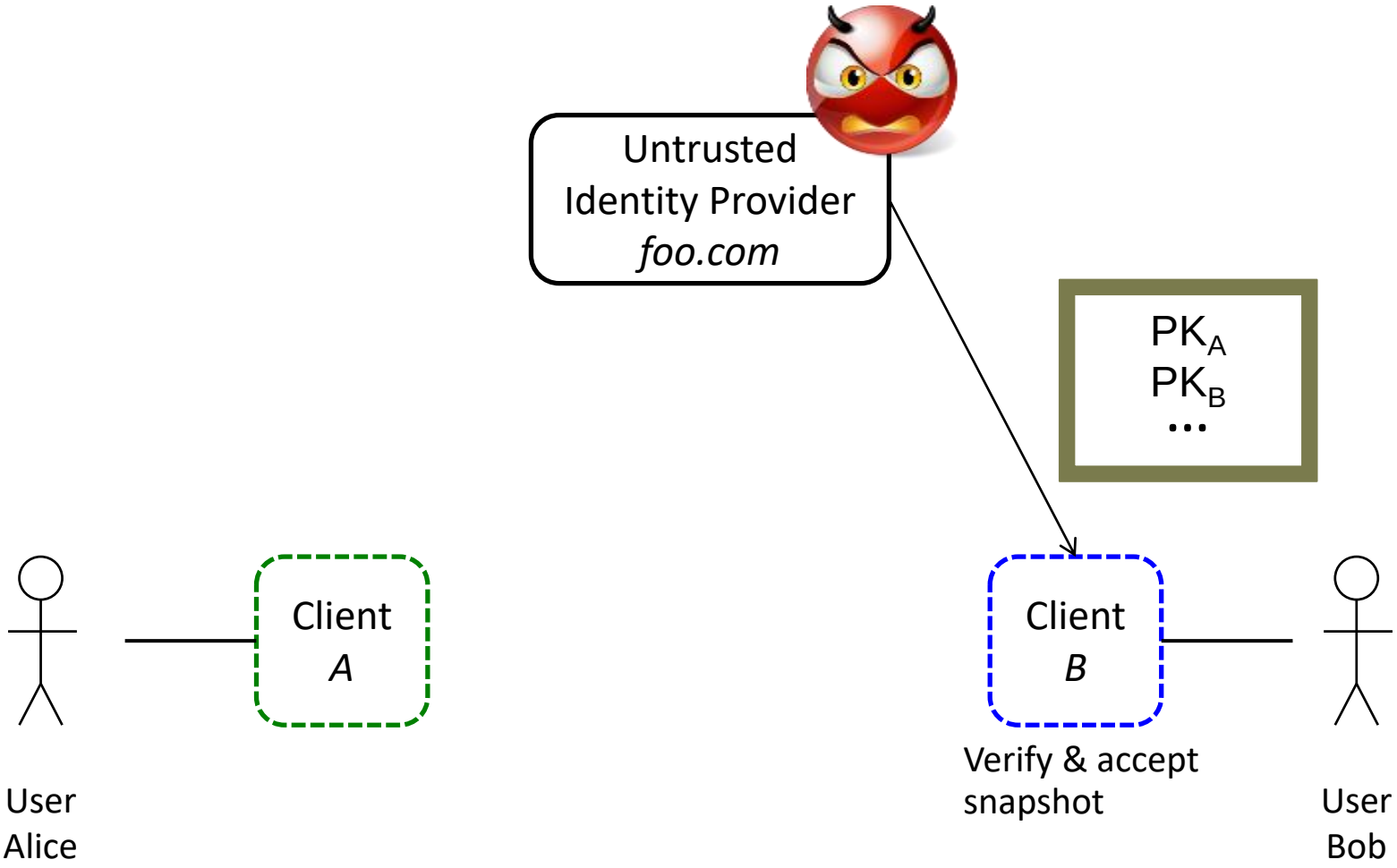
Provider regularly generates and publishes snapshots

Lookups without Inclusion Proofs



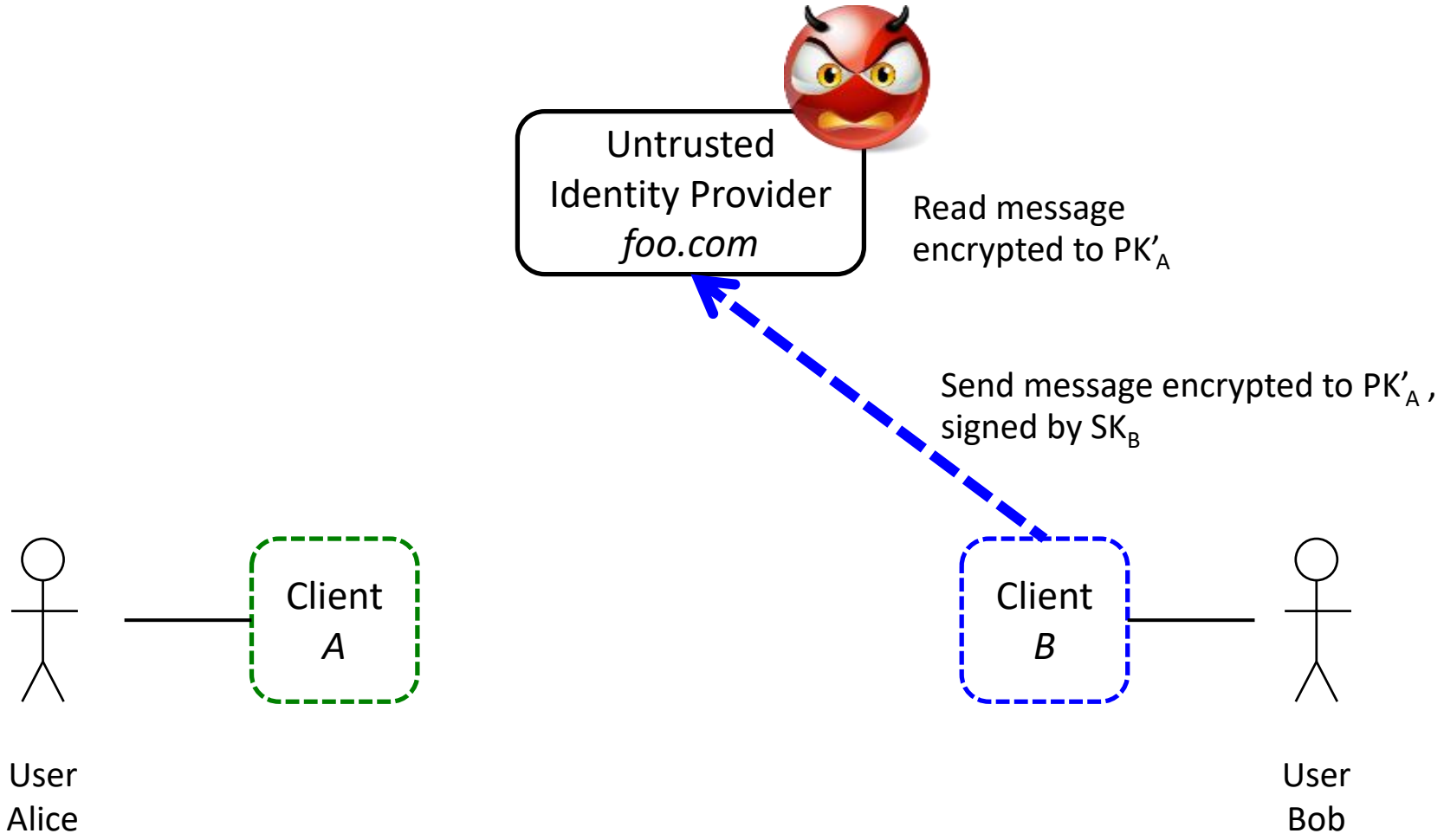
Return Fake Key

Lookups without Inclusion Proofs



No proof fake key is inconsistent with snapshot

Lookups without Inclusion Proofs

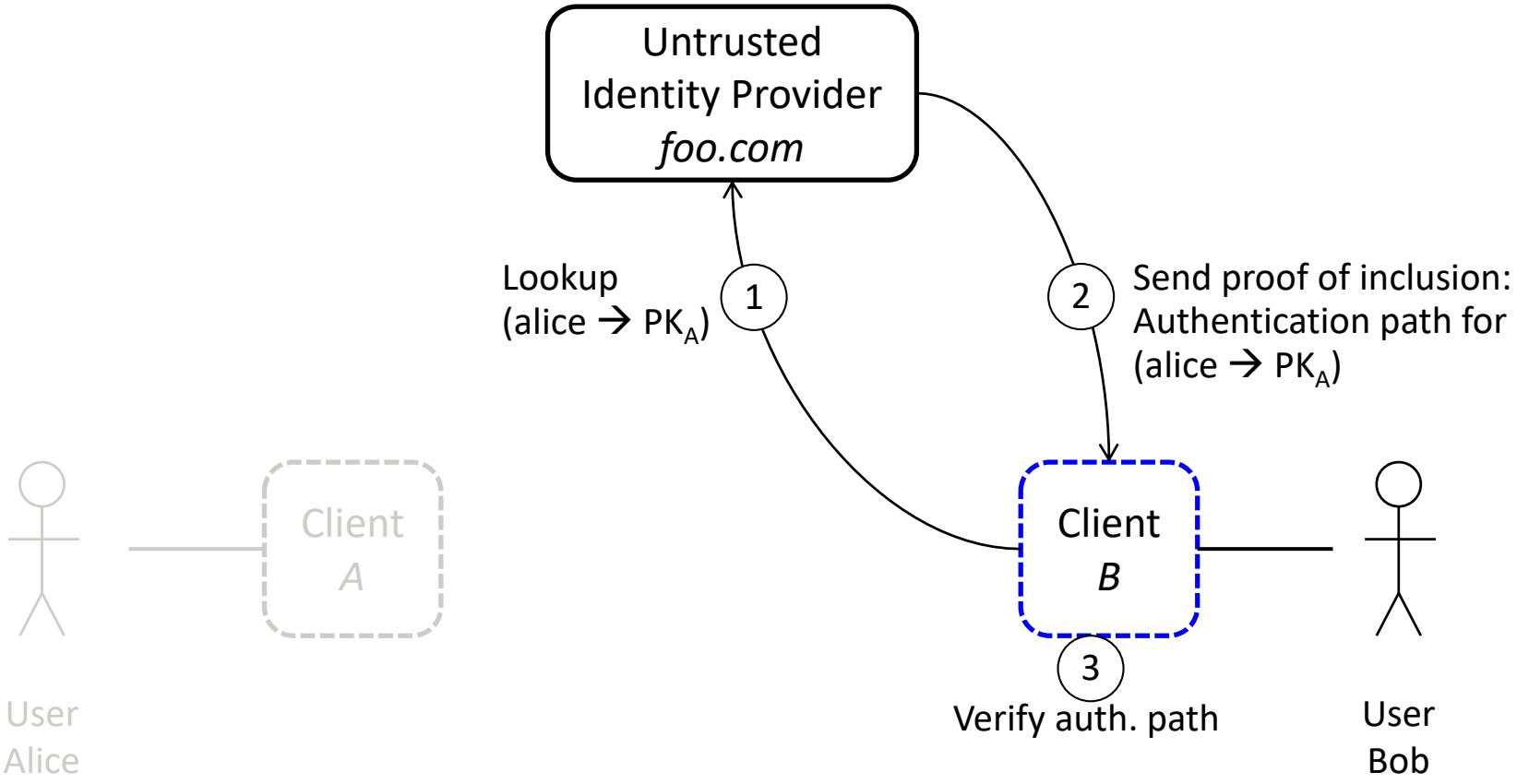


Provider can read Bob's message

Lookup Protocol

- Ensures bindings are consistent with snapshots.
- Prevents malicious providers from publishing fake keys and not leaving any evidence of the misbehavior.

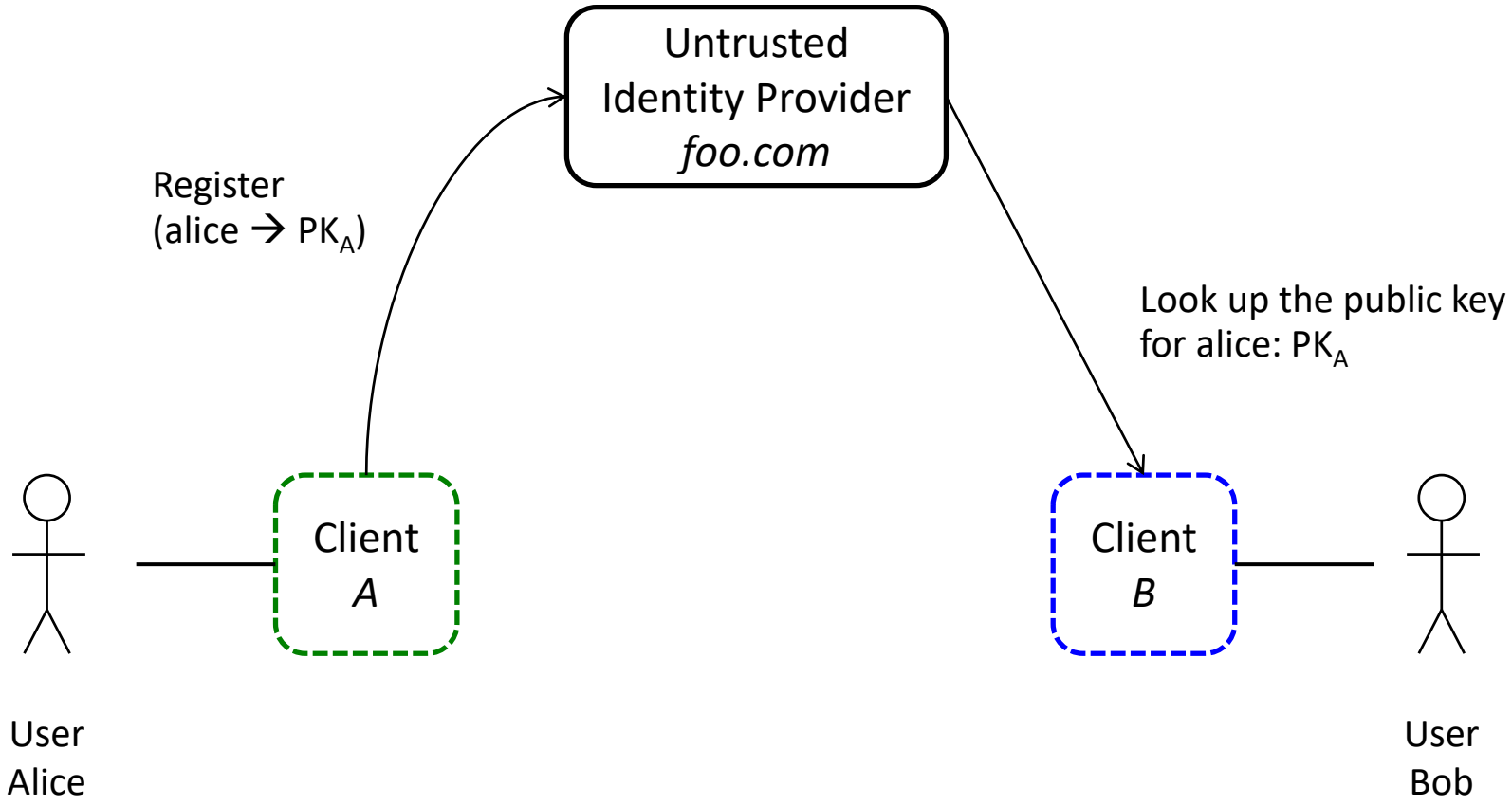
Lookup Protocol



CONIKS Protocols

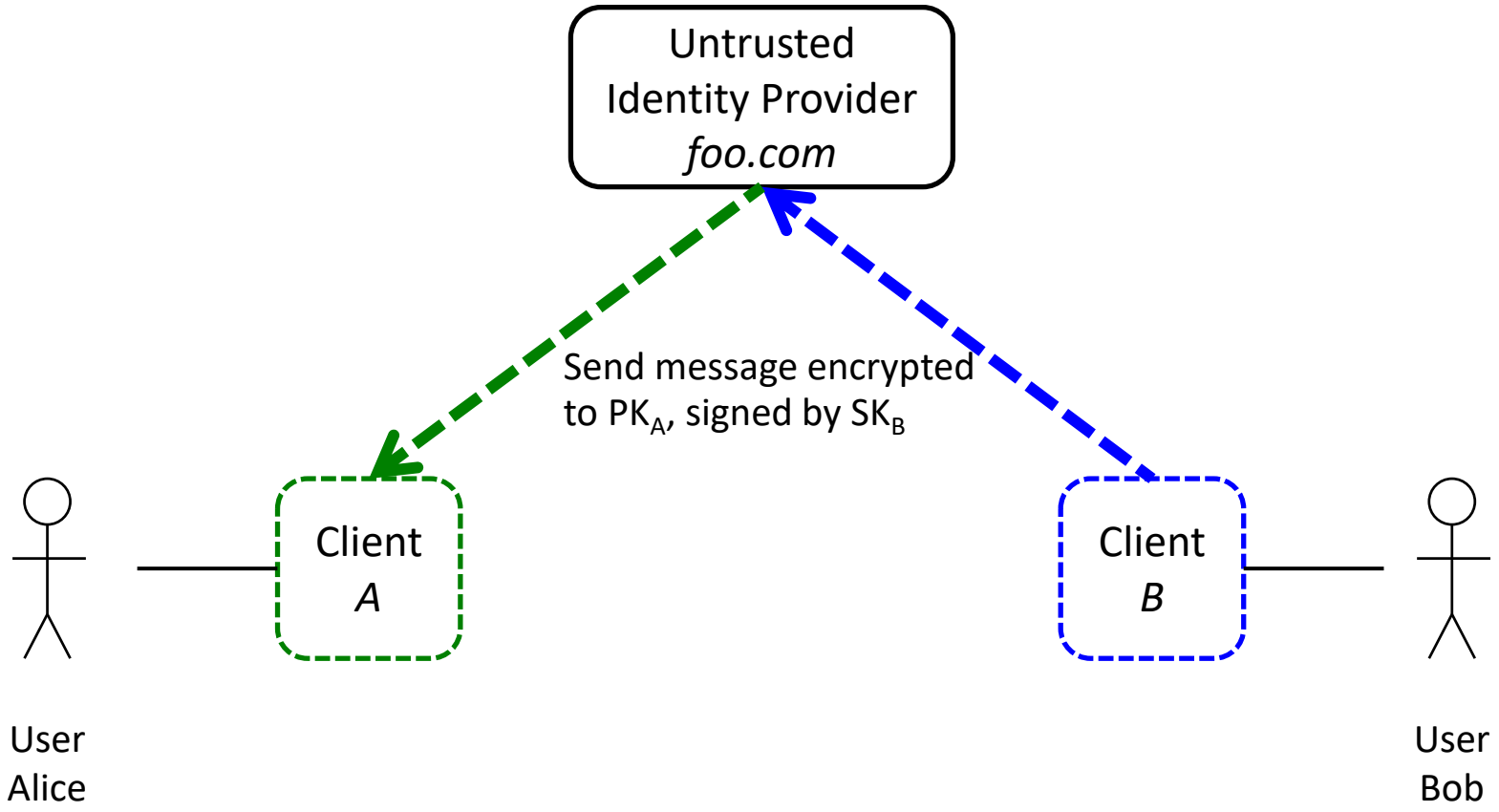
- Registration: new bindings/updated keys.
- Lookups: Clients check bindings are included in directory.
- Monitoring: Clients check for validity of bindings.
- Auditing: Clients & providers check for non-equivocation.

Communication without Monitoring



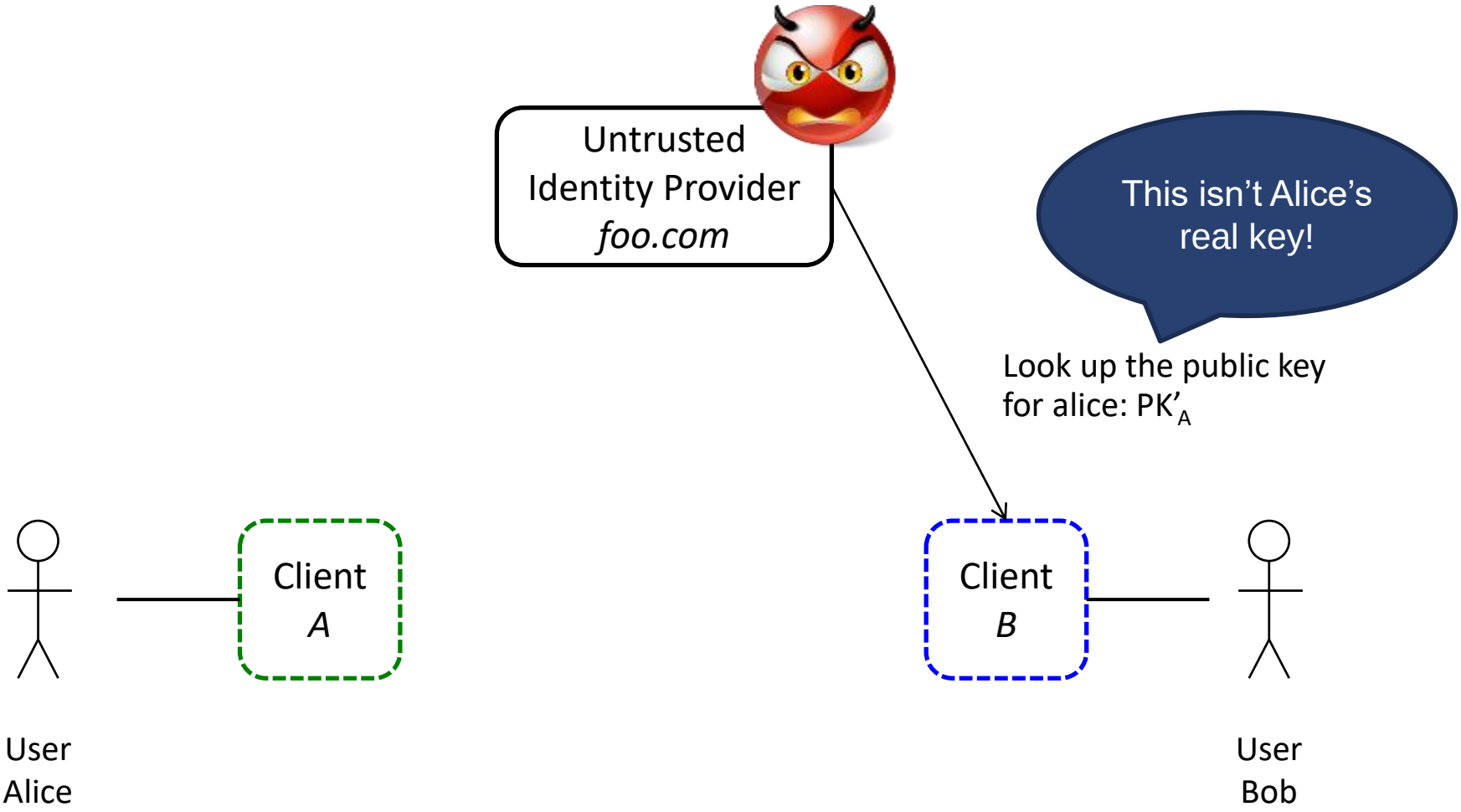
Epoch 1: Key has not Changed

Communication without Monitoring



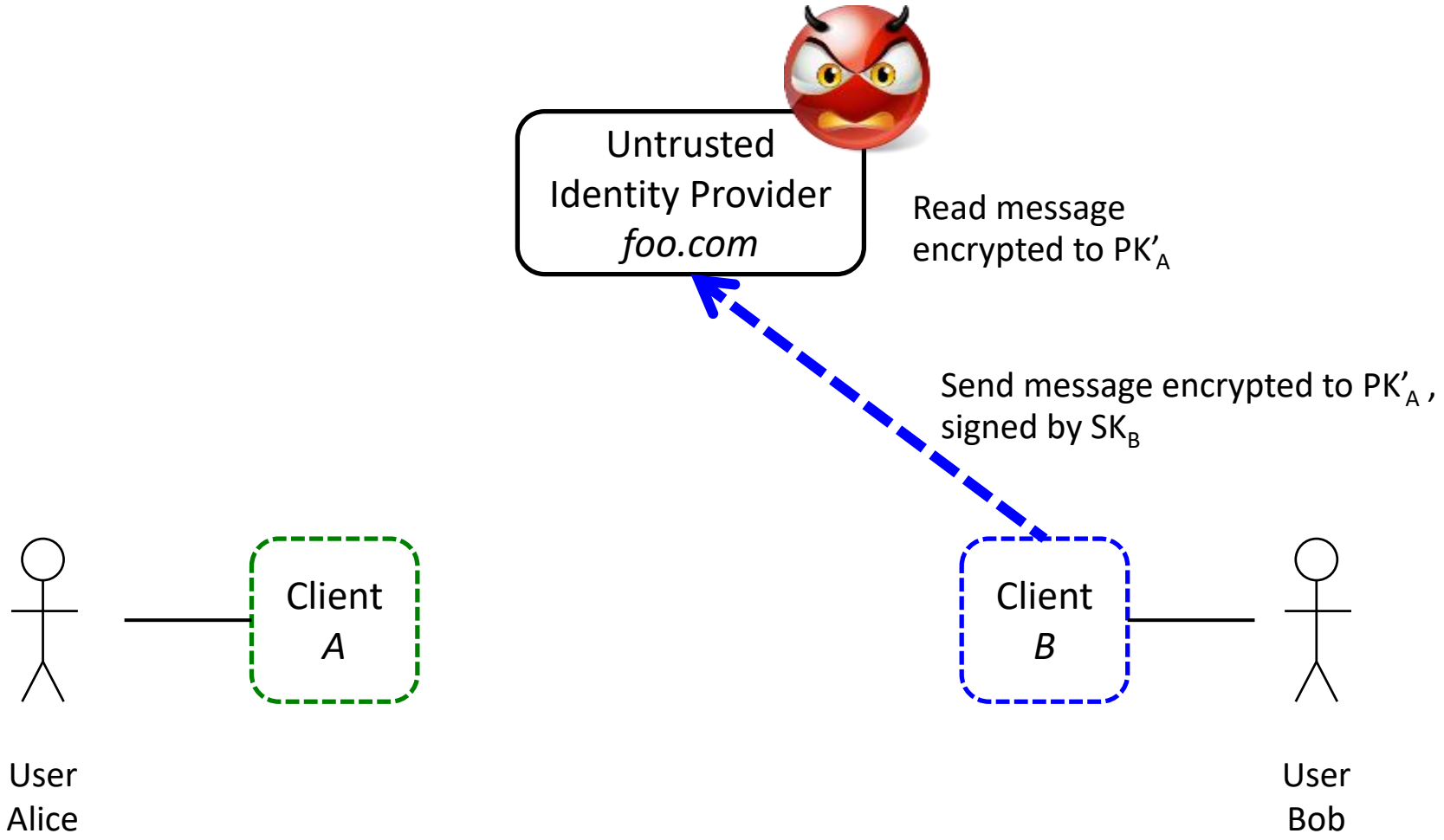
Epoch 1: Provider cannot read Bob's message

Communication without Monitoring



Epoch 2: Key has been Changed

Communication without Monitoring

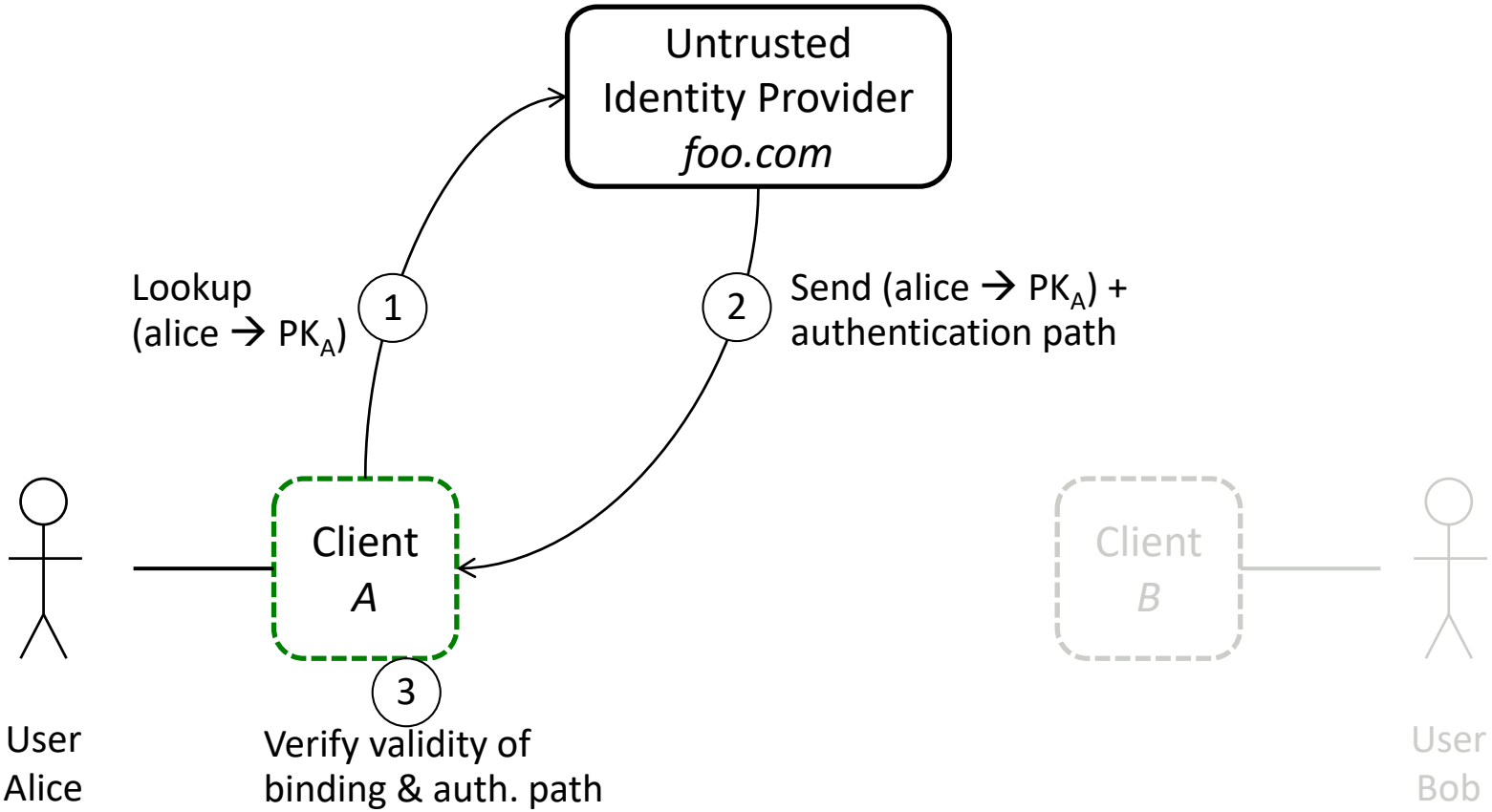


Epoch 2: Provider can read Bob's message

Monitoring Protocol

- Ensures public keys do not change unexpectedly.
- Prevents malicious providers from getting away with replacing existing keys.

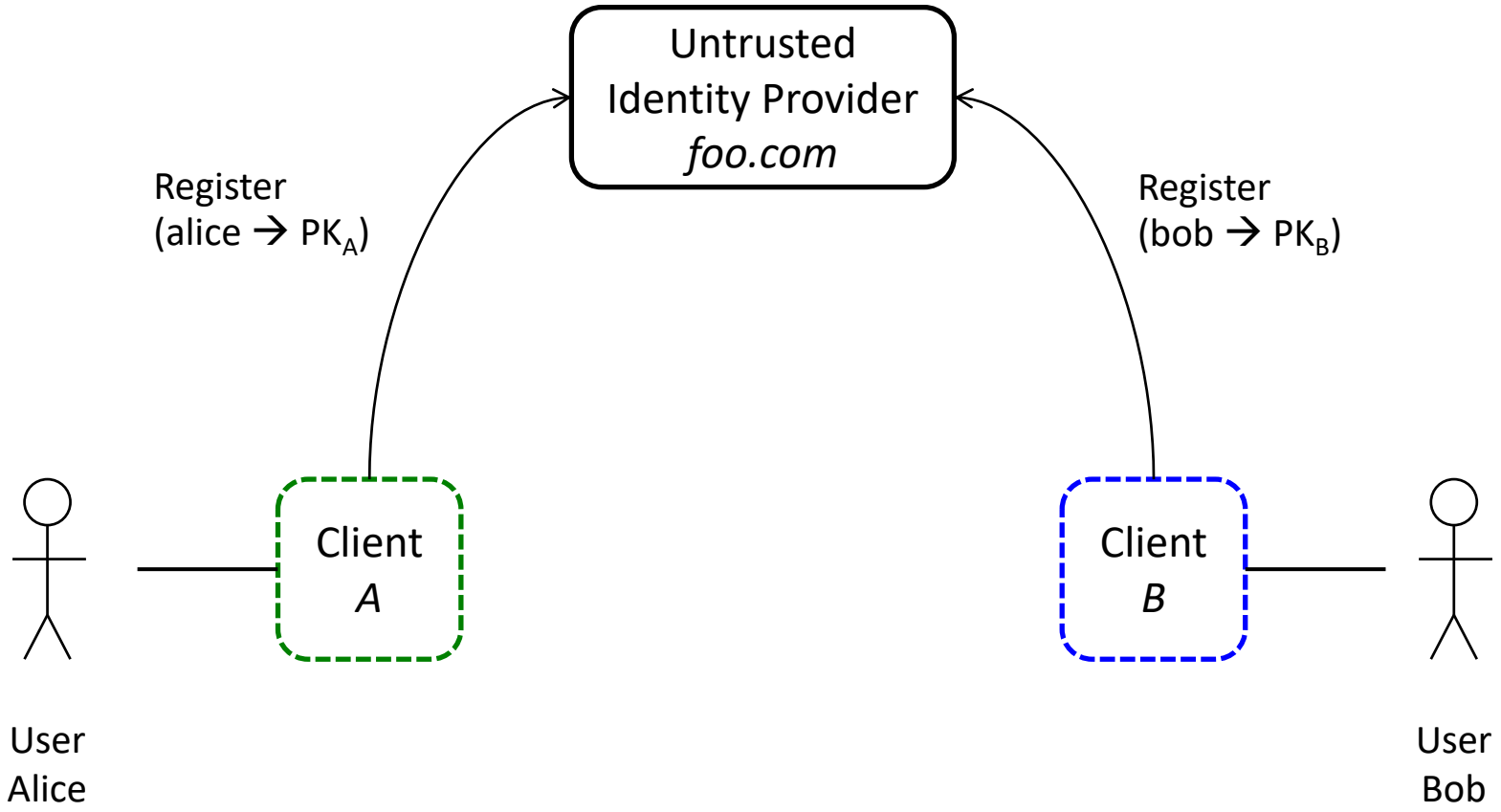
Monitoring Protocol



CONIKS Protocols

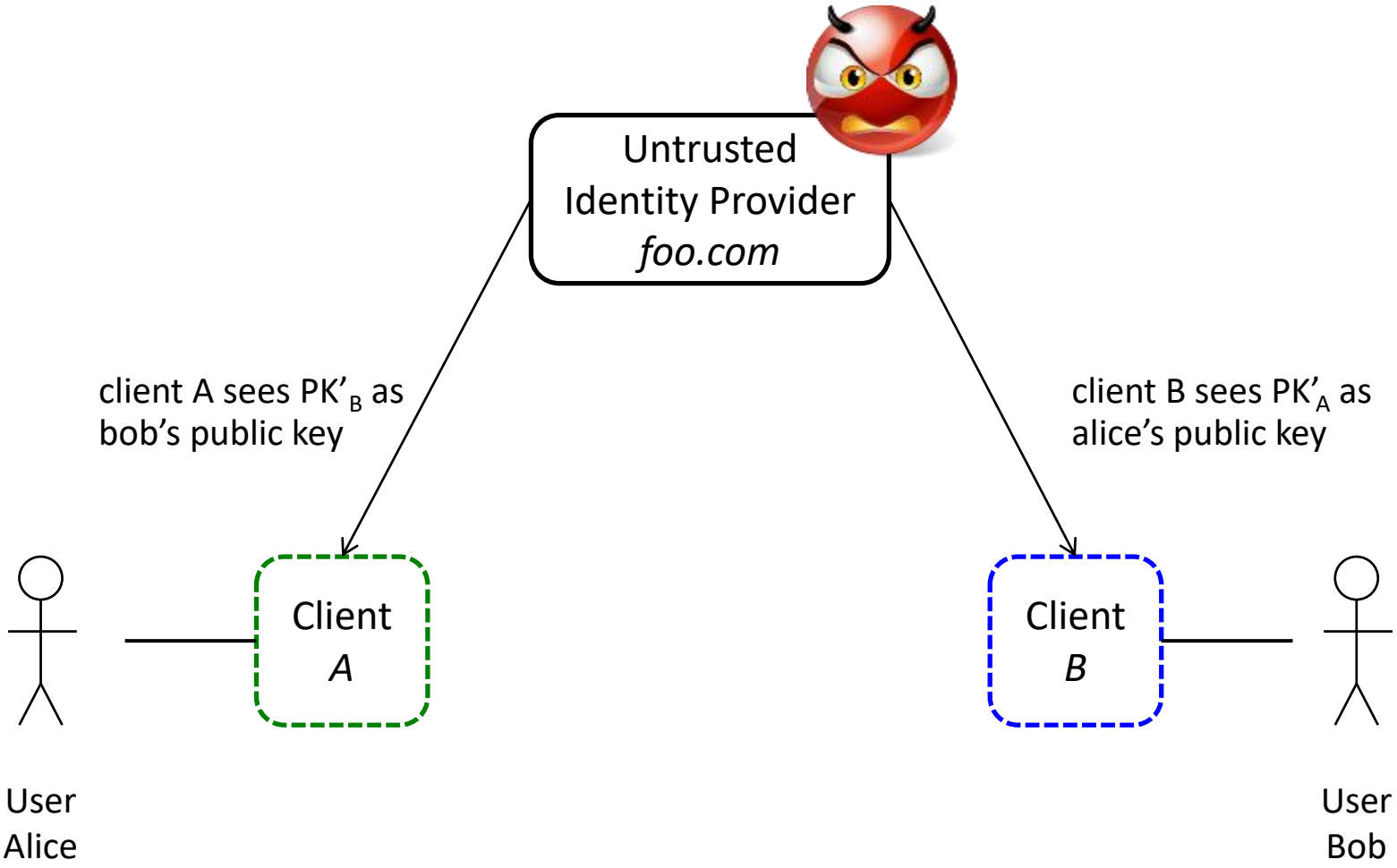
- Registration: new bindings/updated keys.
- Lookups: Clients check bindings are included in directory.
- Monitoring: Clients check for validity of bindings.
- Auditing: Clients & providers check for non-equivocation.

Communication without Auditing



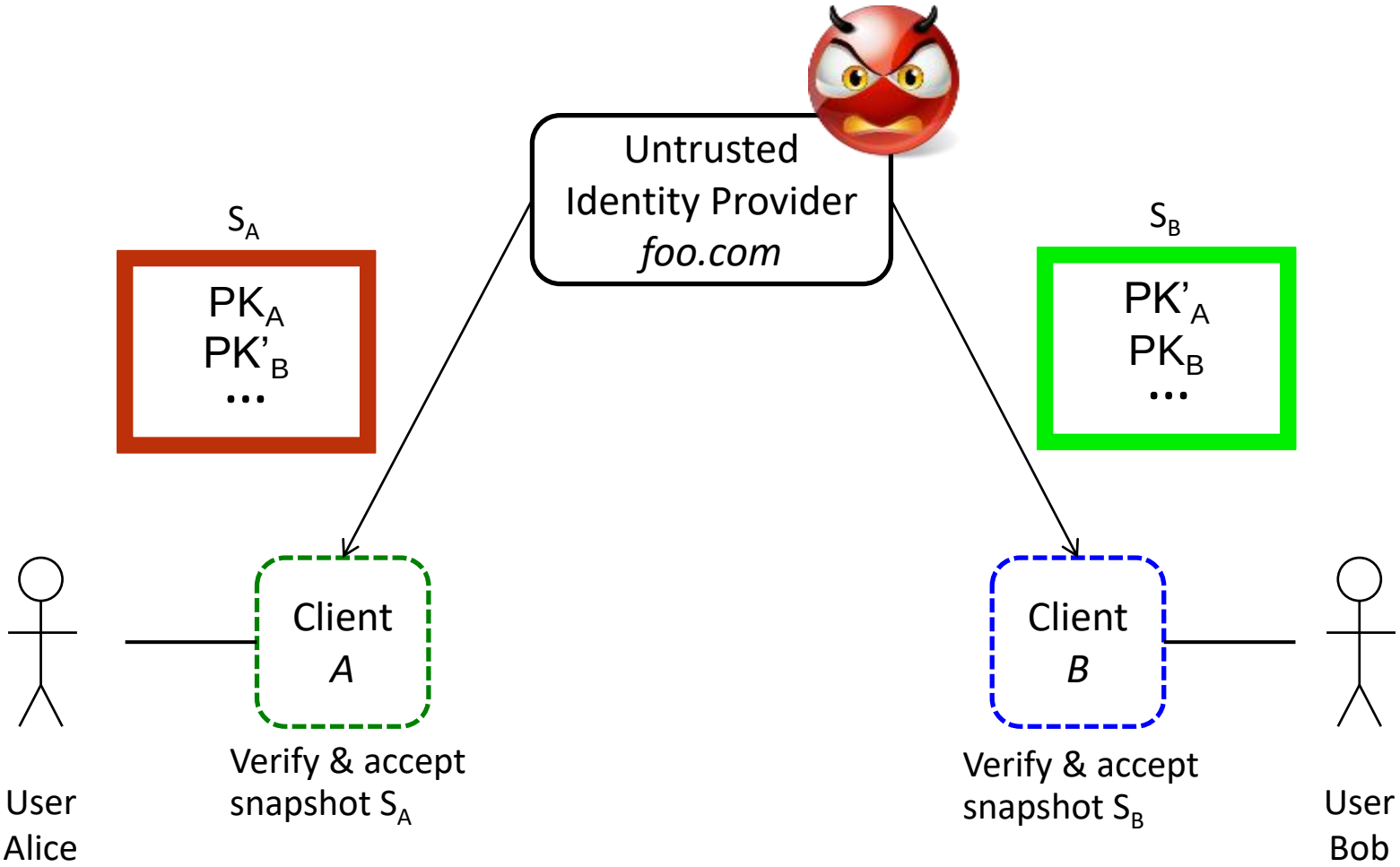
Clients register legitimate Keys

Communication without Auditing



Provider creates two versions of its Directory

Communication without Auditing

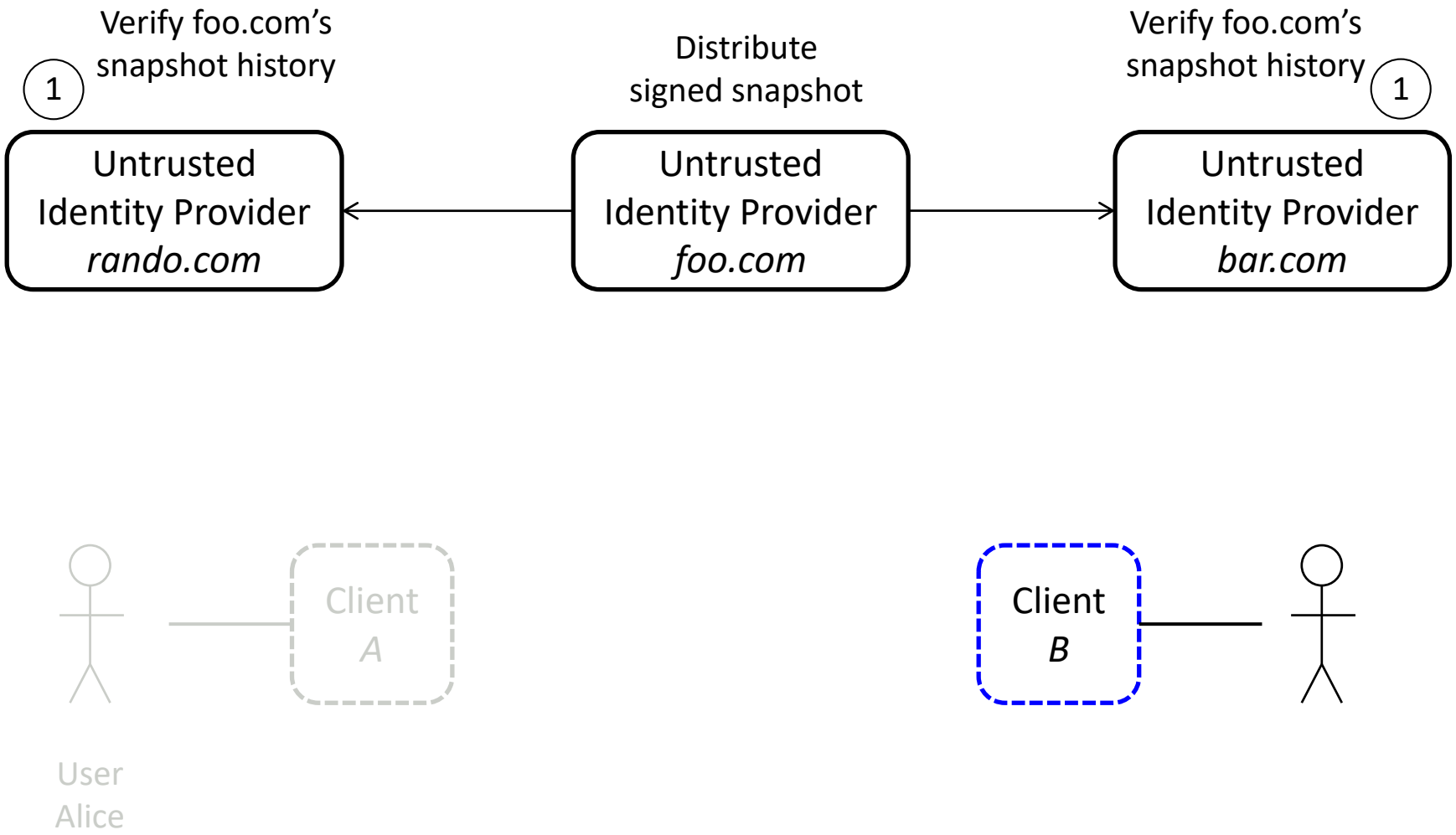


Provider presents two different but valid snapshots

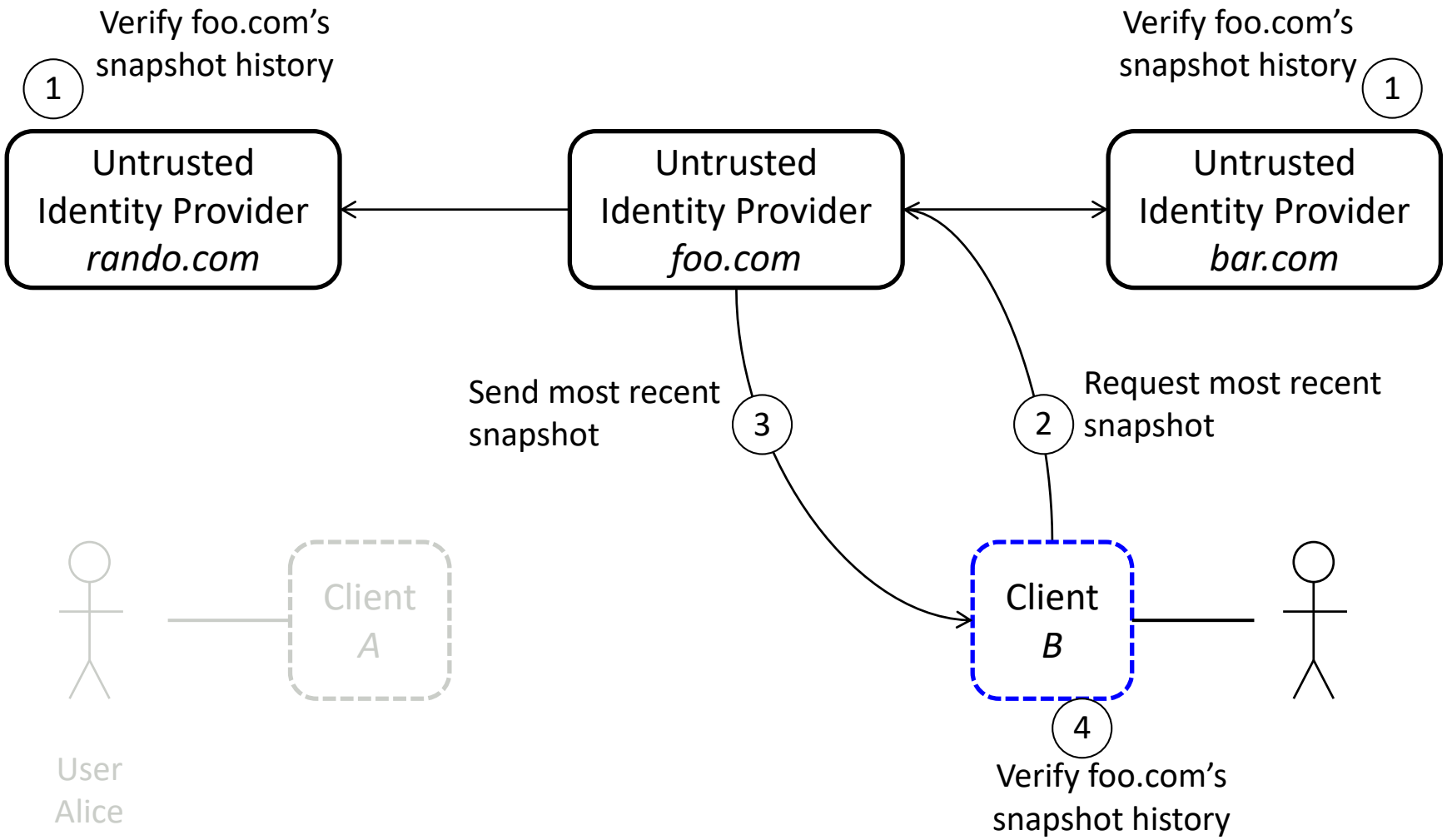
Auditing Protocol

- Ensures clients have consistent views of directories.
- Clients query random other providers for snapshots observed from given provider.
- Prevents malicious providers from getting away with publishing inconsistent versions of key directory.

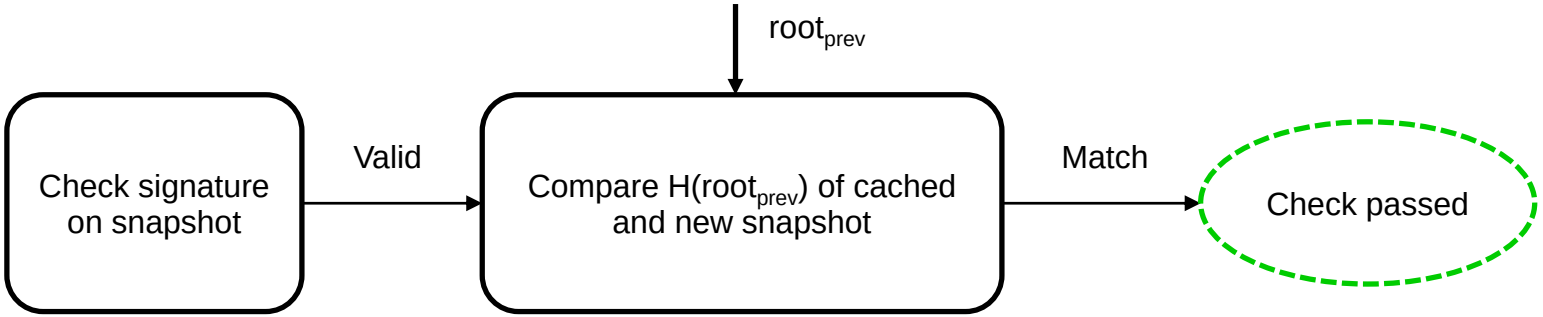
Auditing Protocol



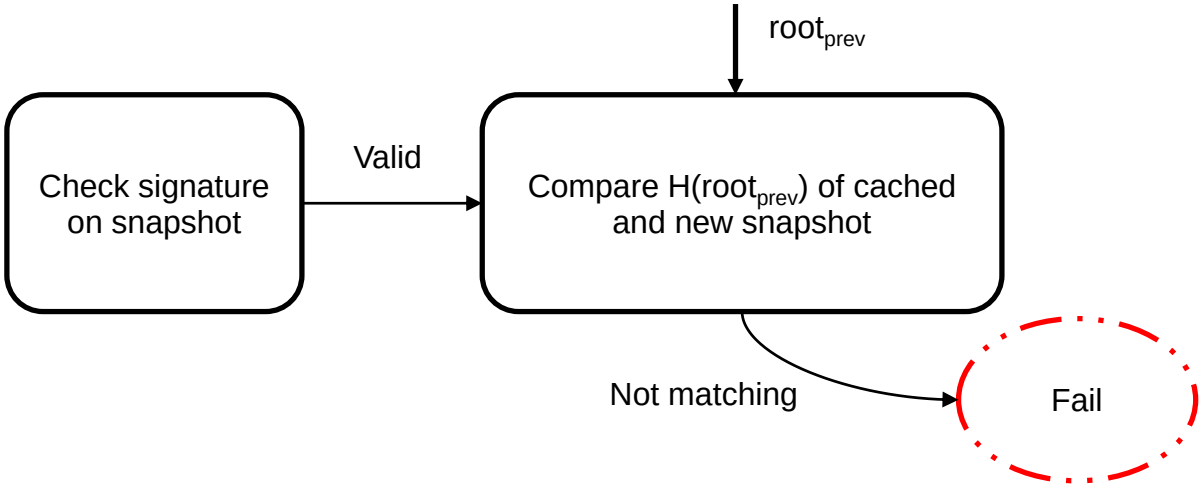
Auditing Protocol



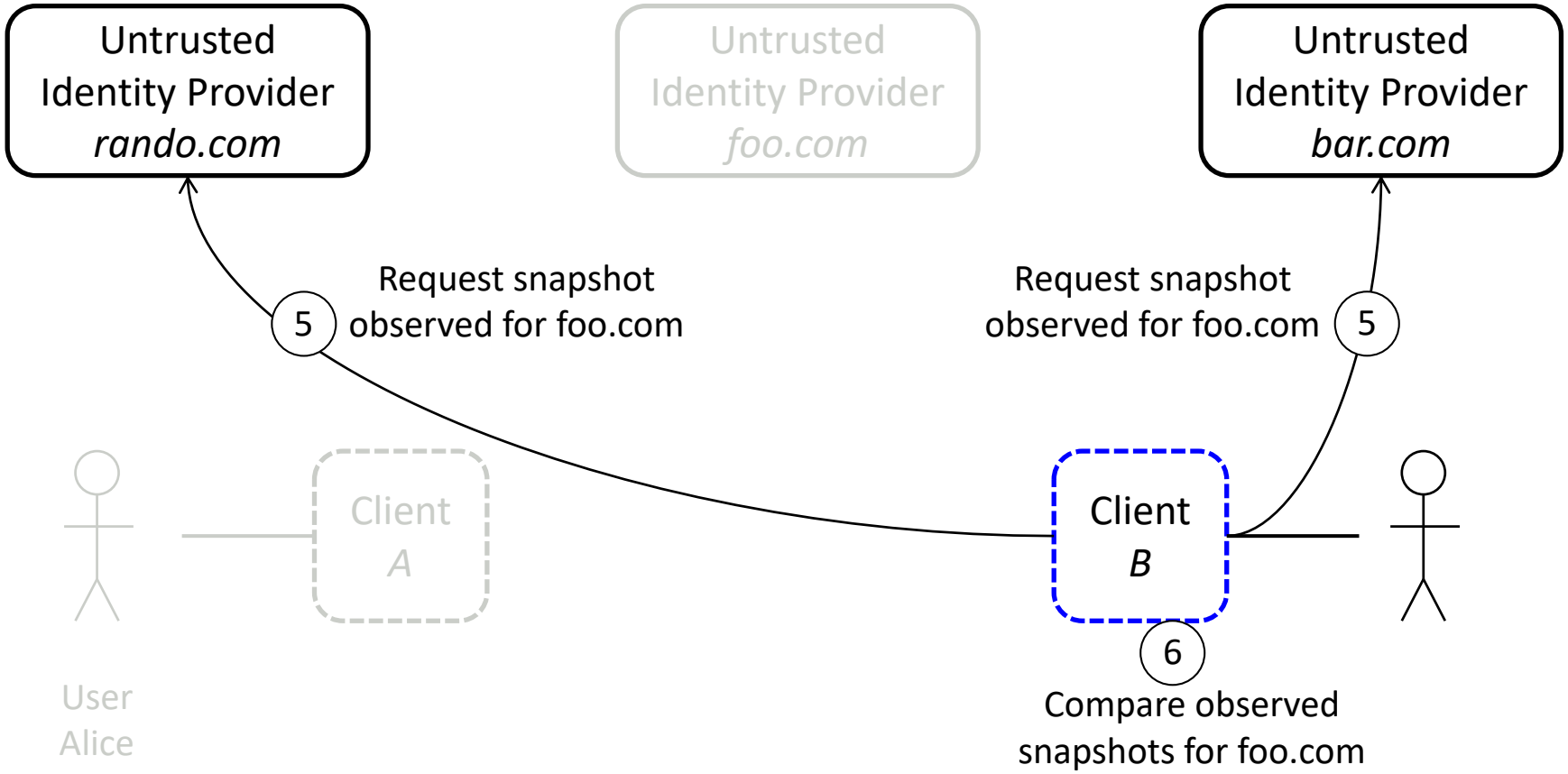
Checking Snapshot History



Checking Snapshot History



Auditing Protocol



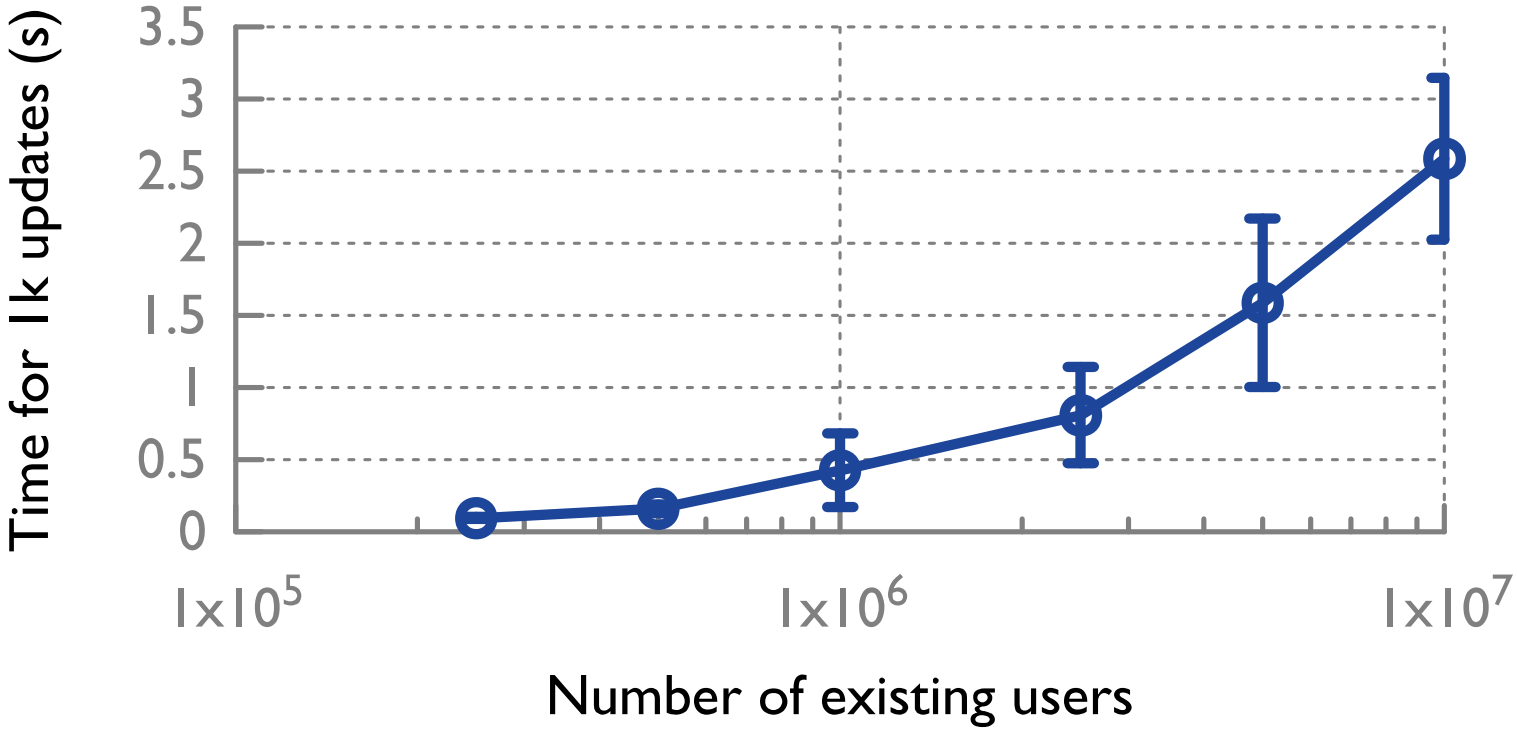
Privacy problems with this protocol?

- Auditors see contents of directory
- Authentication paths reveal other entries in the directory

Prototype Implementation

- CONIKS Chat: Secure chat service.
- Stand-alone provider alongside XMPP server.
- Supports registration, lookups and basic auditing.

Server performance



Client Overheads

	Download Requirements	**Storage Requirements**
Lookup (per binding)	< 1.4KB	0B
Monitoring (epoch)	< 800B	~ 300B
Monitoring (day)	< 20KB	~ 300B
Auditing (epoch, per snapshot)	~ 100B	~ 100B
Auditing (day, per snapshot)	~ 2.5KB	~ 100B

- $N \approx 4$ bill. users, $n \approx 2$ mill. updates/epoch, $d = 24$ epochs/day.

Client Overheads

	Download Requirements	**Storage Requirements**
Lookup (per binding)	< 1.4KB	0B
Monitoring (epoch)	< 800B	~ 300B
Monitoring (day)	< 20KB	~ 300B
Auditing (epoch, per snapshot)	~ 100B	~ 100B
Auditing (day, per snapshot)	~ 2.5KB	~ 100B

- $N \approx 4$ bill. users, $n \approx 2$ mill. updates/epoch, $d = 24$ epochs/day.

Client Overheads

	Download Requirements	**Storage Requirements**
Lookup (per binding)	< 1.4KB	~ 300B
Monitoring (epoch)	< 800B	~ 300B
Monitoring (day)	< 20KB	~ 300B
Auditing (epoch, per snapshot)	~ 100B	~ 100B
Auditing (day, per snapshot)	~ 2.5KB	~ 100B

- $N \approx 4$ bill. users, $n \approx 2$ mill. updates/epoch, $d = 24$ epochs/day.

Client Overheads

	Download Requirements	**Storage Requirements**
Lookup (per binding)	< 1.4KB	~ 300B
Monitoring (epoch)	< 800B	~ 300B
Monitoring (day)	< 20KB	~ 300B
Auditing (epoch, per snapshot)	~ 100B	~ 100B
Auditing (day, per snapshot)	~ 2.5KB	~ 100B

- $N \approx 4$ bill. users, $n \approx 2$ mill. updates/epoch, $d = 24$ epochs/day.

CONIKS summary

- CONIKS is an efficiently verifiable, privacy-preserving key management service for end-user public keys.
- End-to-end security can be practical for end-users.
- Some industry adoption: e.g. Google’s Key Transparency