Efficient Key Authentication Service for Secure End-to-end Communications

KOÇ UNIVERSITY

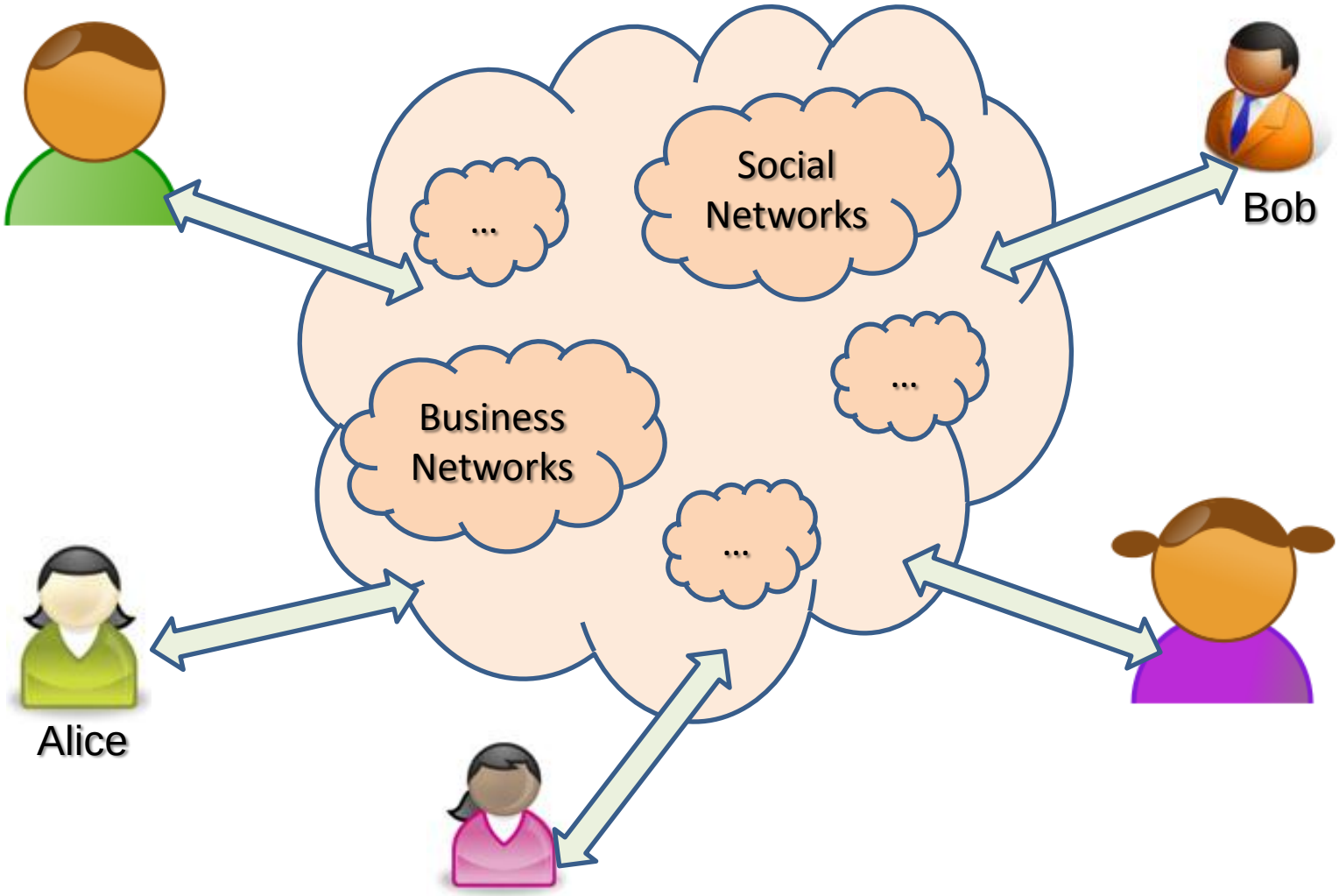
Mohammad Etemad

Alptekin Küpçü

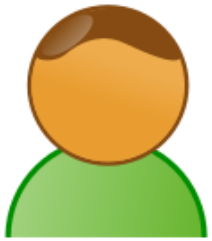
Outline

- Overview
  - Problem definition
  - Certificate authority
- Certificate transparency
  - Existing solutions
  - Definition and problems
- Our solution
  - Overview and properties
  - Performance analysis comparison to previous work
- Conclusion
- References

Overview



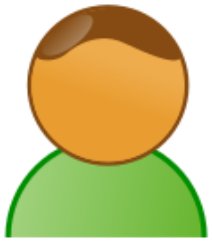
Overview



Bob

Alice

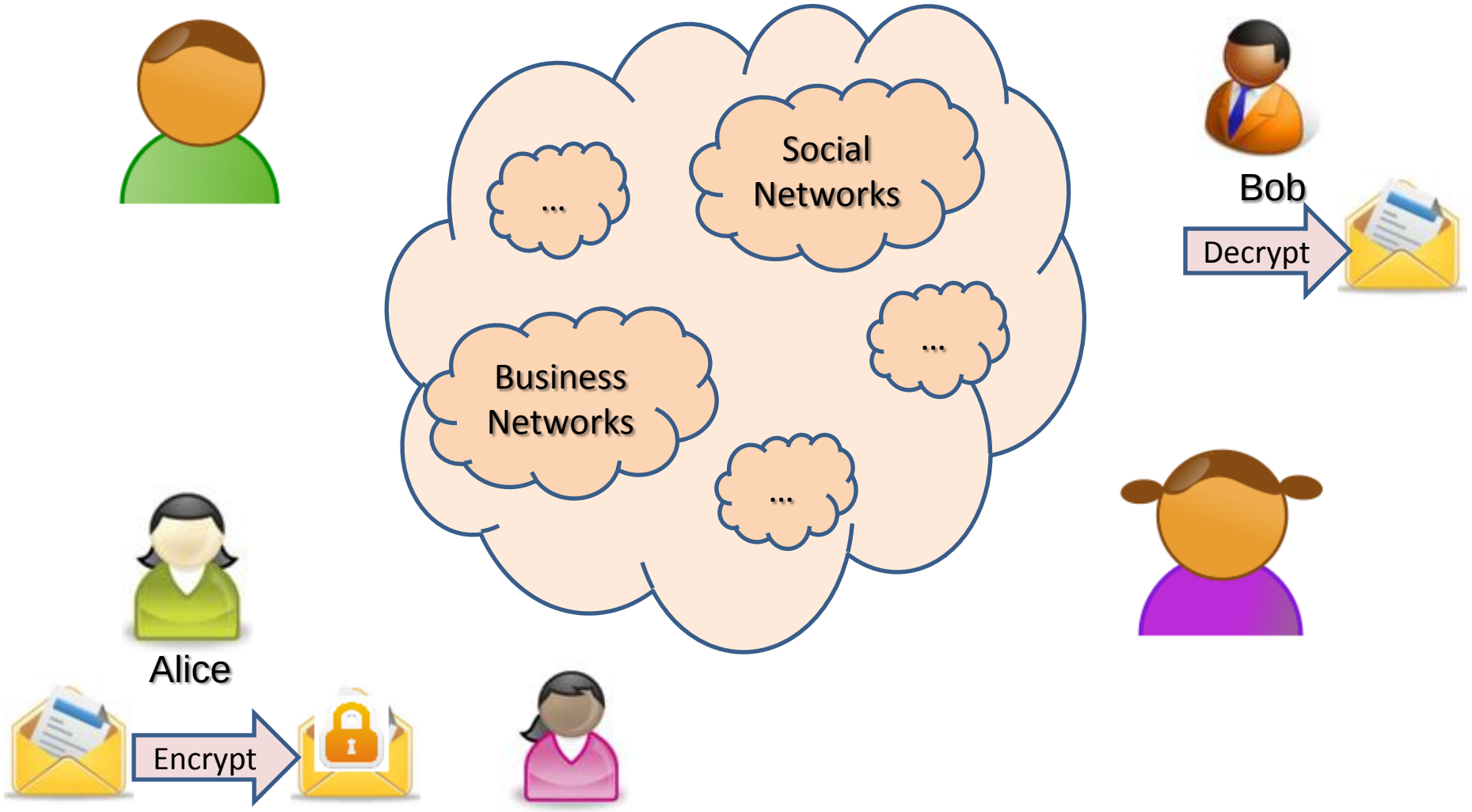
Overview



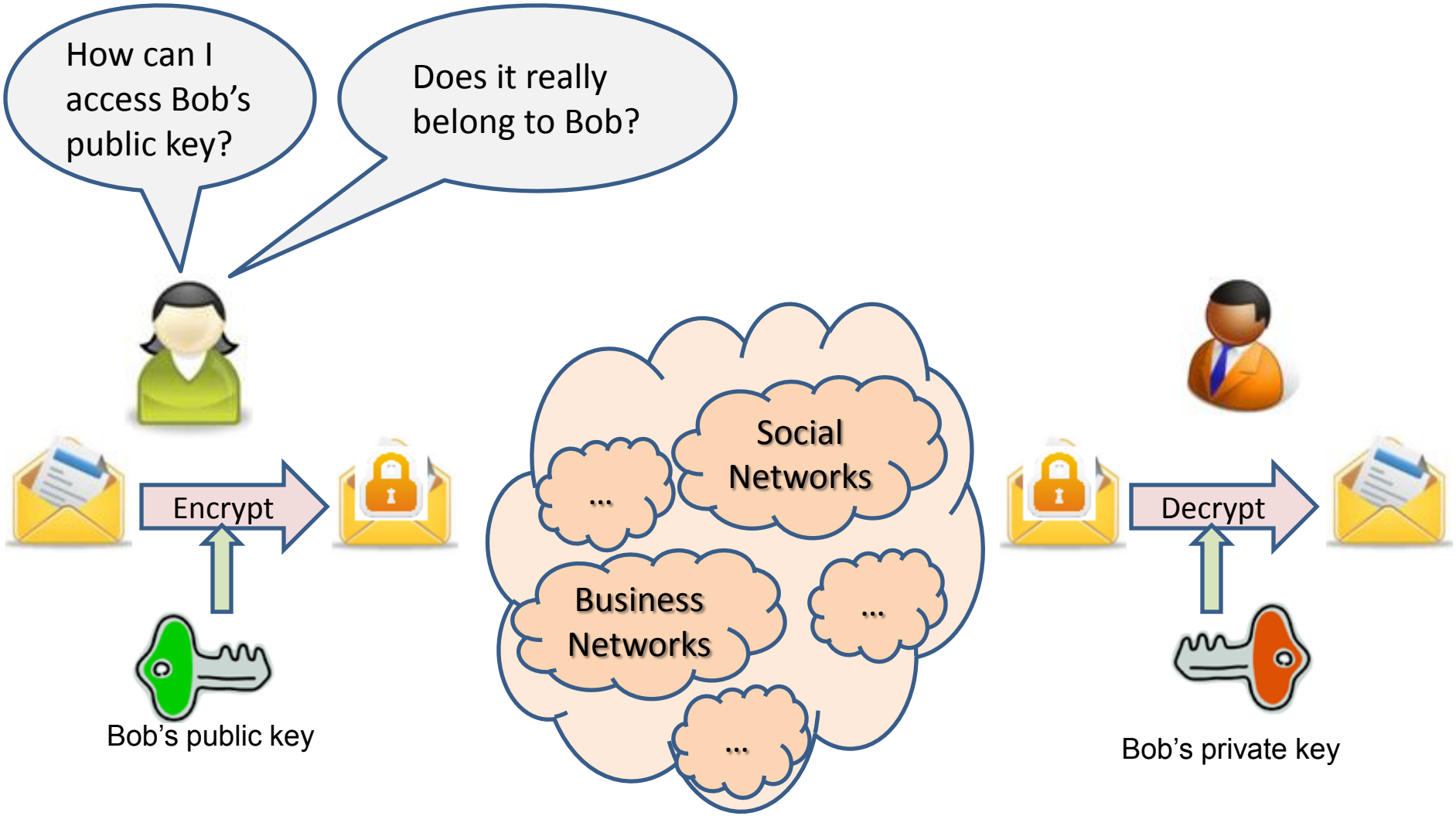
Bob

Alice

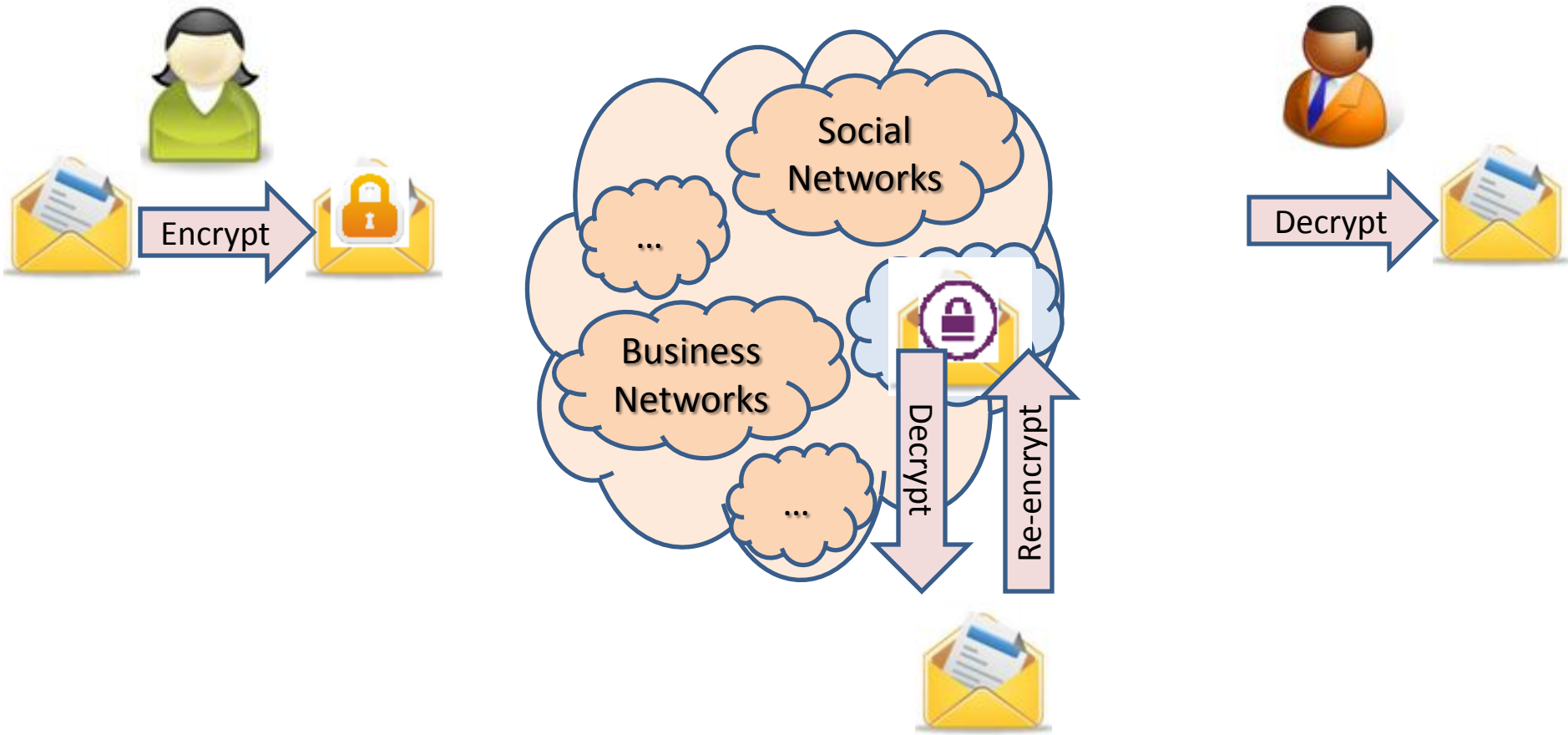
Overview


...
Social Networks
Business Networks
...
...
Bob
Decrypt
Alice
Encrypt

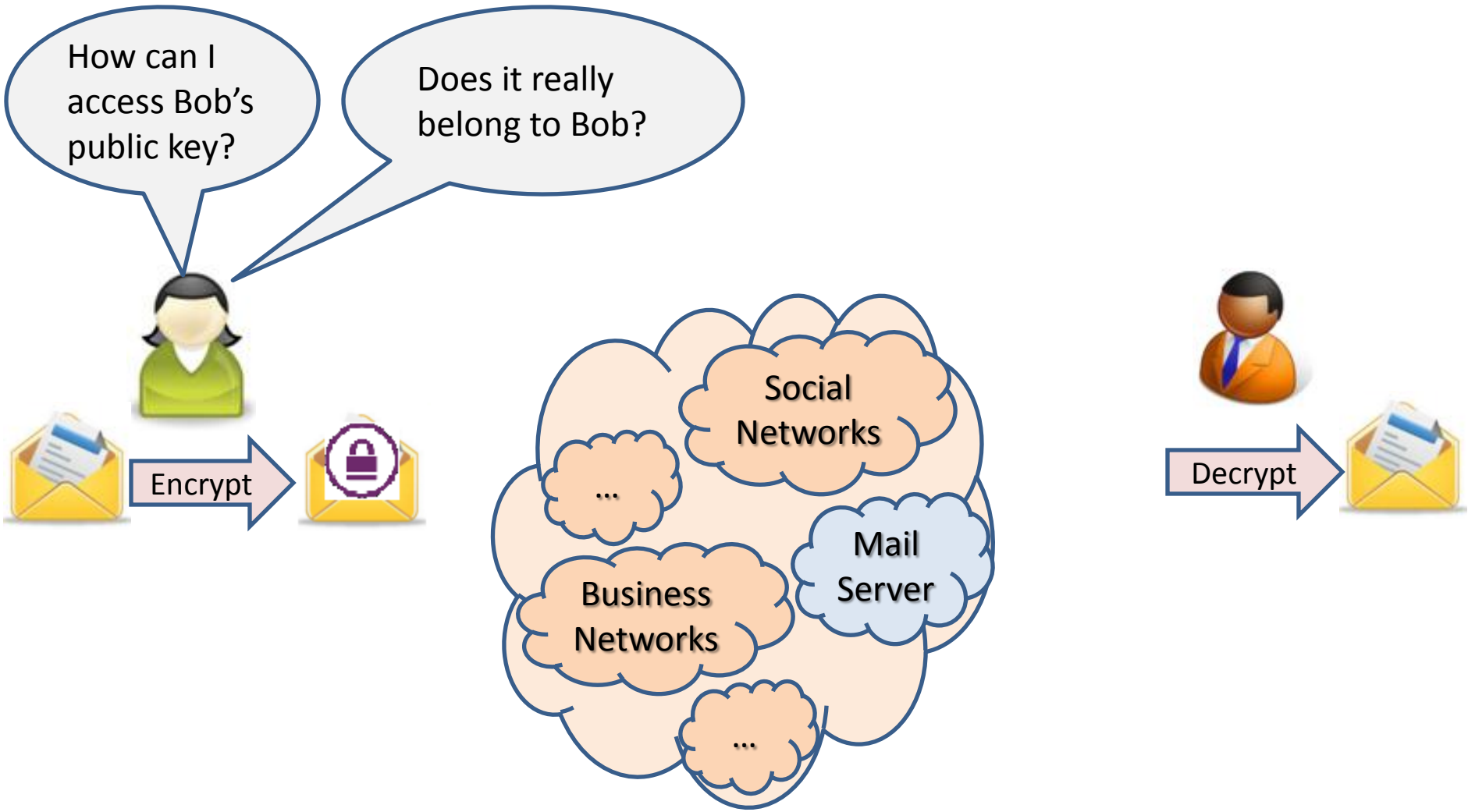
The problem


How can I access Bob's public key?
Does it really belong to Bob?
Encrypt
Bob's public key
Social Networks
...
Business Networks
...
...
Decrypt
Bob's private key

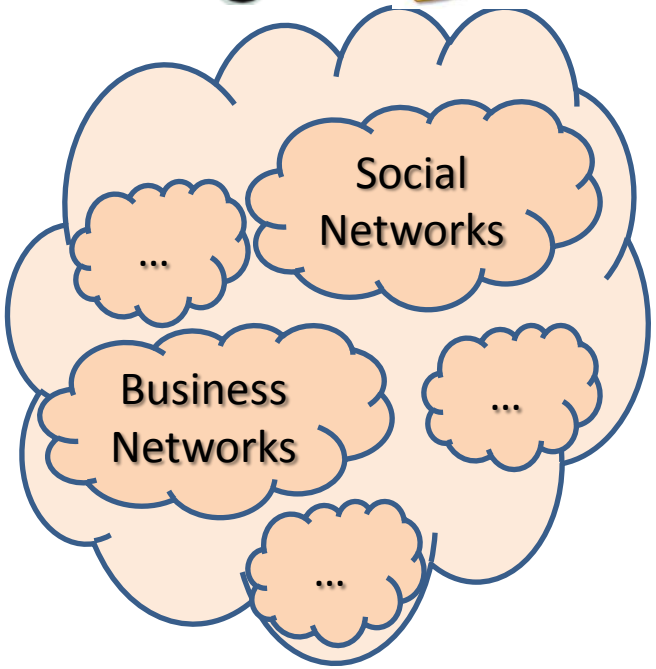
Point-to-point Encryption



End-to-end Encryption



Certificate Authority



Certificate Transparency

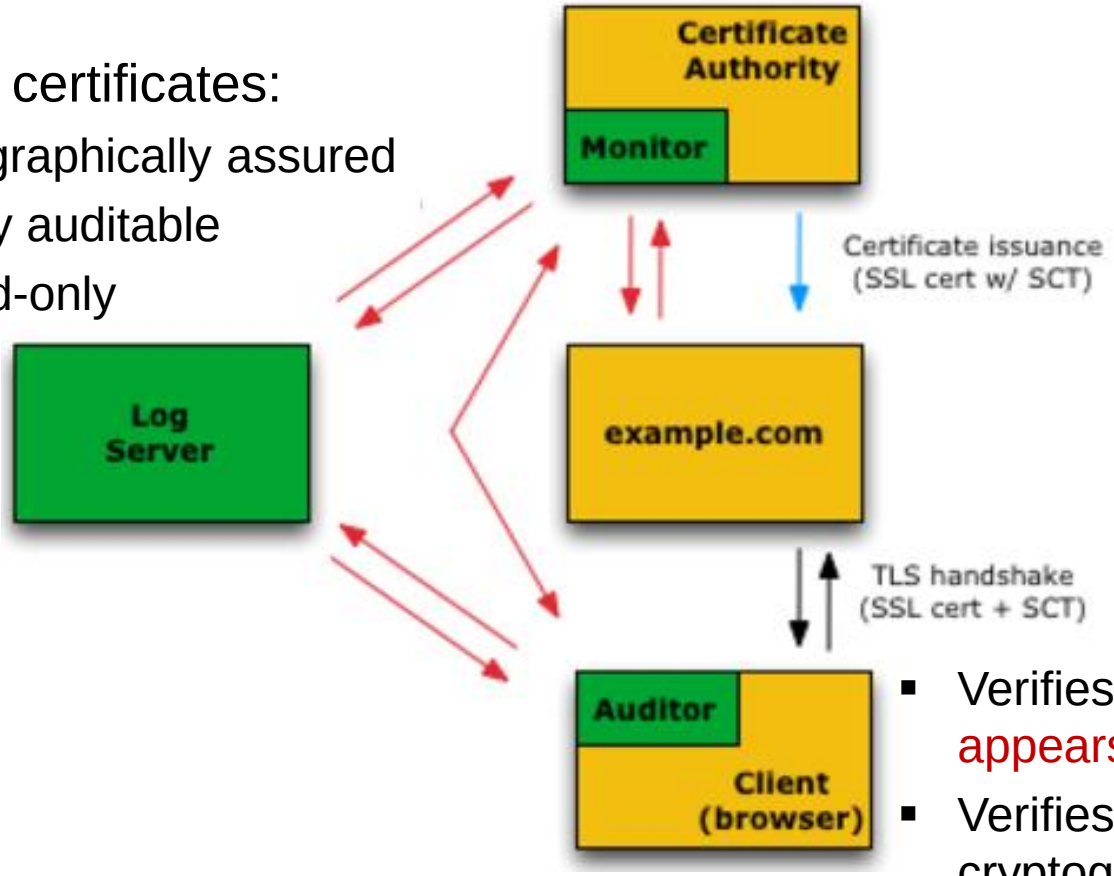
- It takes a while to find, report and revoke a fraudulent certificate.
- Certificate transparency proposed by Google recently [LLK13]:
  - A user (domain) regularly checks the certificates issued on her name.
  - In case of any misbehavior, reports (and publishes) it.
  - Other users rely on the fact that any misbehavior should have already been caught by the key owner.
- No need to trust a third party.

Certificate Transparency

(https://www.certificate-transparency.org/)

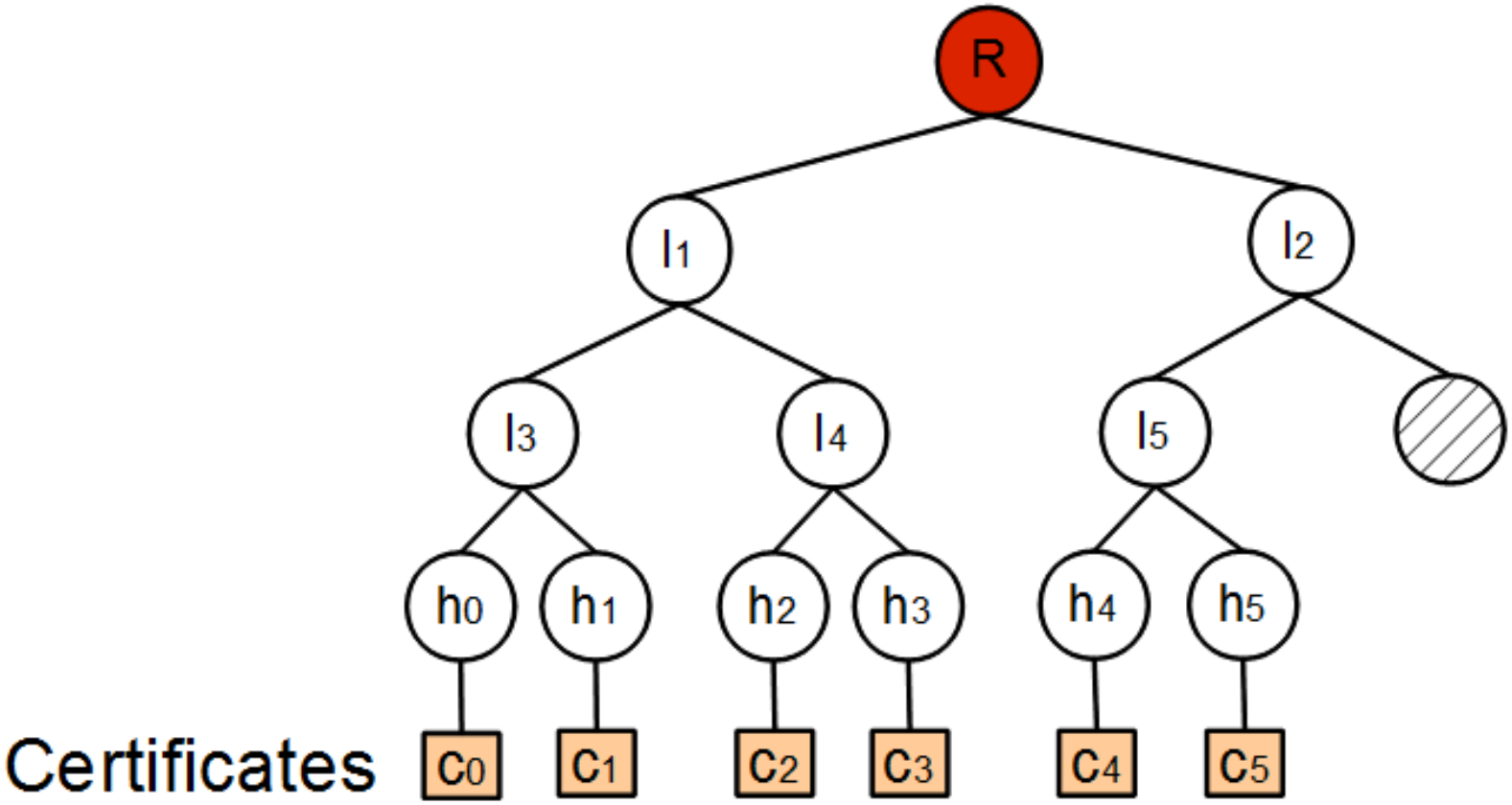
- Are public servers
- Run periodically
- Look for suspicious certificates

- Maintains certificates:
  - cryptographically assured
  - publicly auditable
  - append-only

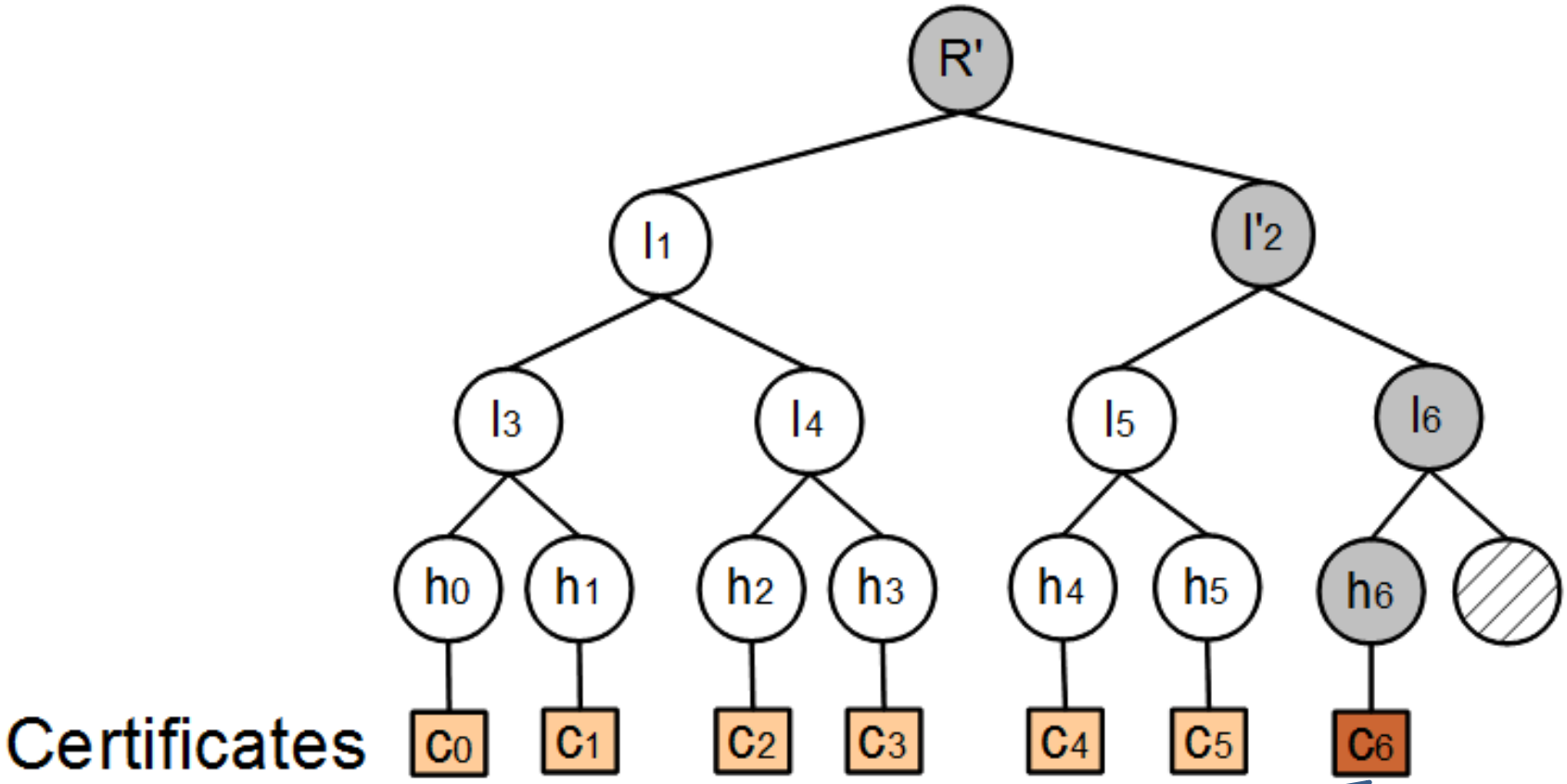


- Verifies a particular certificate appears in a log
- Verifies logs are cryptographically consistent

Certificate Transparency



Certificate Transparency



Enhanced Certificate Transparency

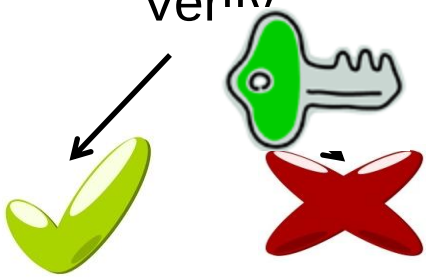
- Problems in certificate transparency
  - All certificate owners should check and make sure they have not been affected by any update.
  - Revocation cost is *O(n), n* is the number of registered certificates.
  - Client-side gossiping requires a large communication, not efficient

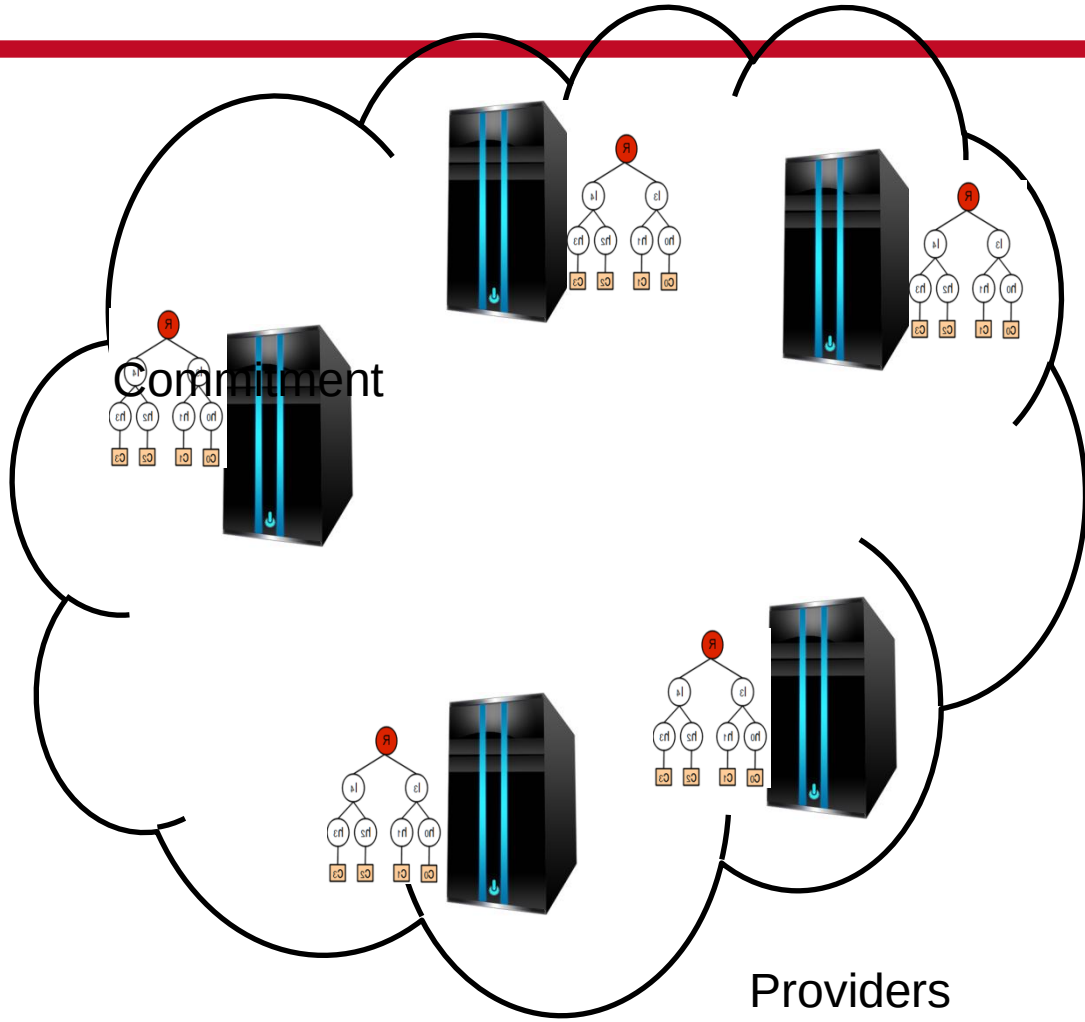
- Enhanced certificate transparency [MR14]
  - Reduces the revocation complexity from *O(n)* to *O(log n)*.

CONIKS

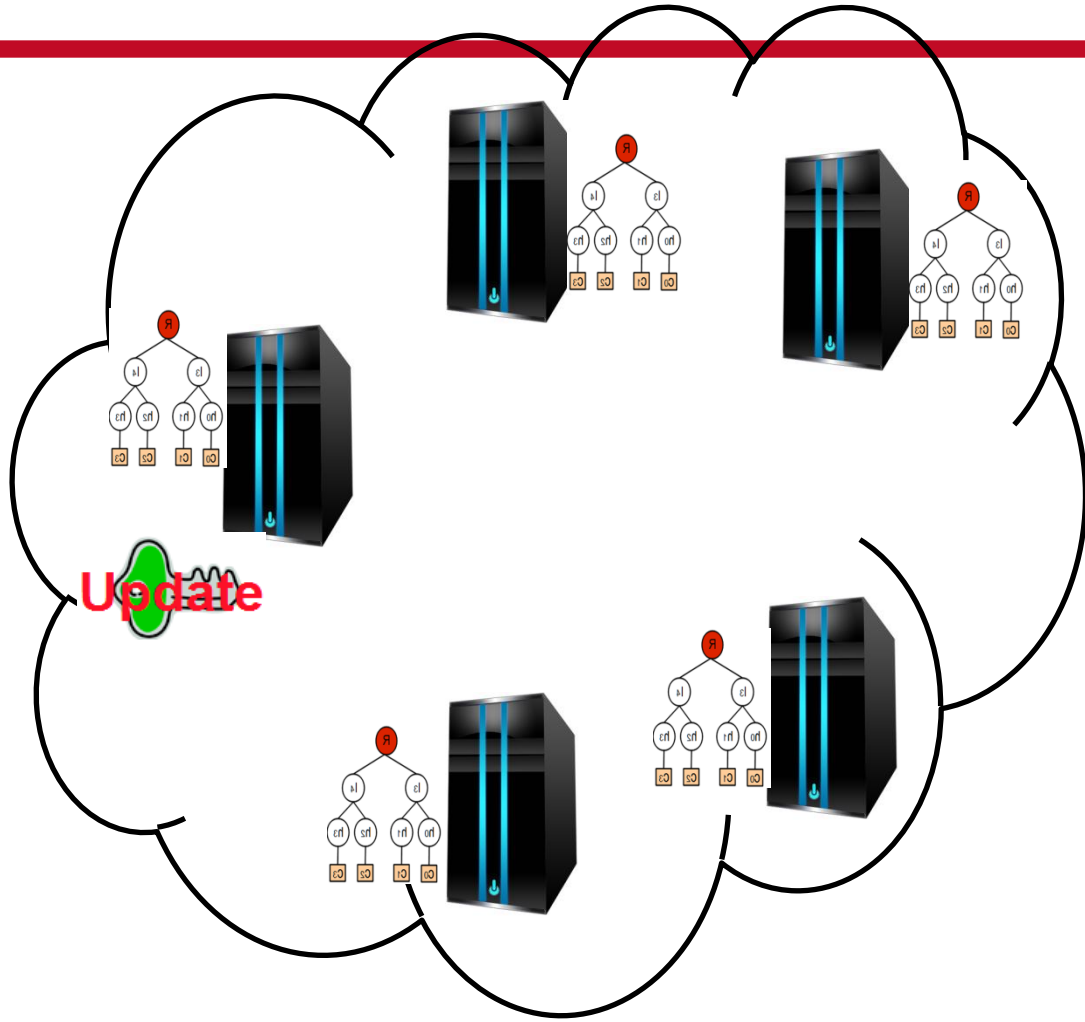
- CONsistent Identity and Key Service [MBF14]
  - An automated key management system.
  - A number of key providers storing users keys.
  - Server-side gossiping.
  - The users can detect equivocations or unexpected key changes.
  - The clients perform checks on epochs.

CONIKS





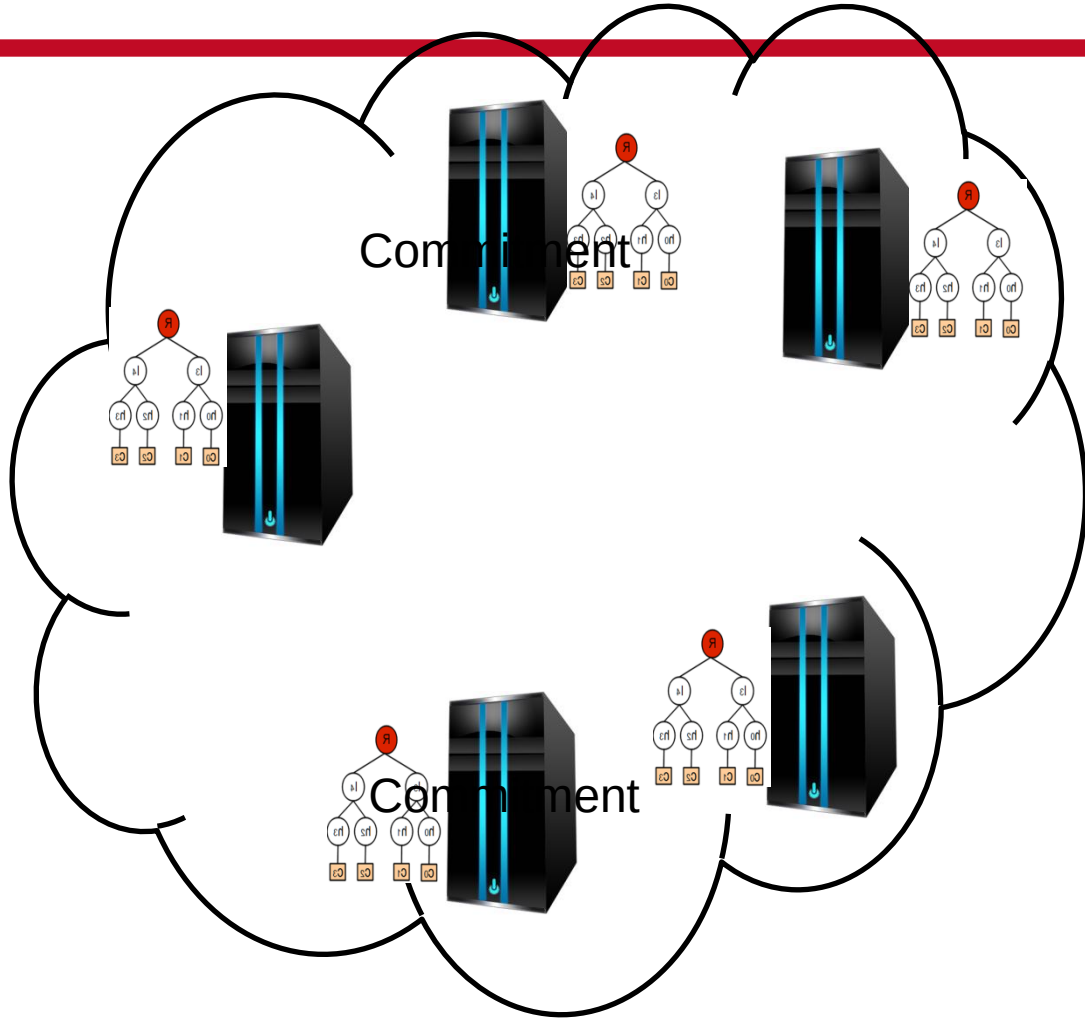
CONIKS



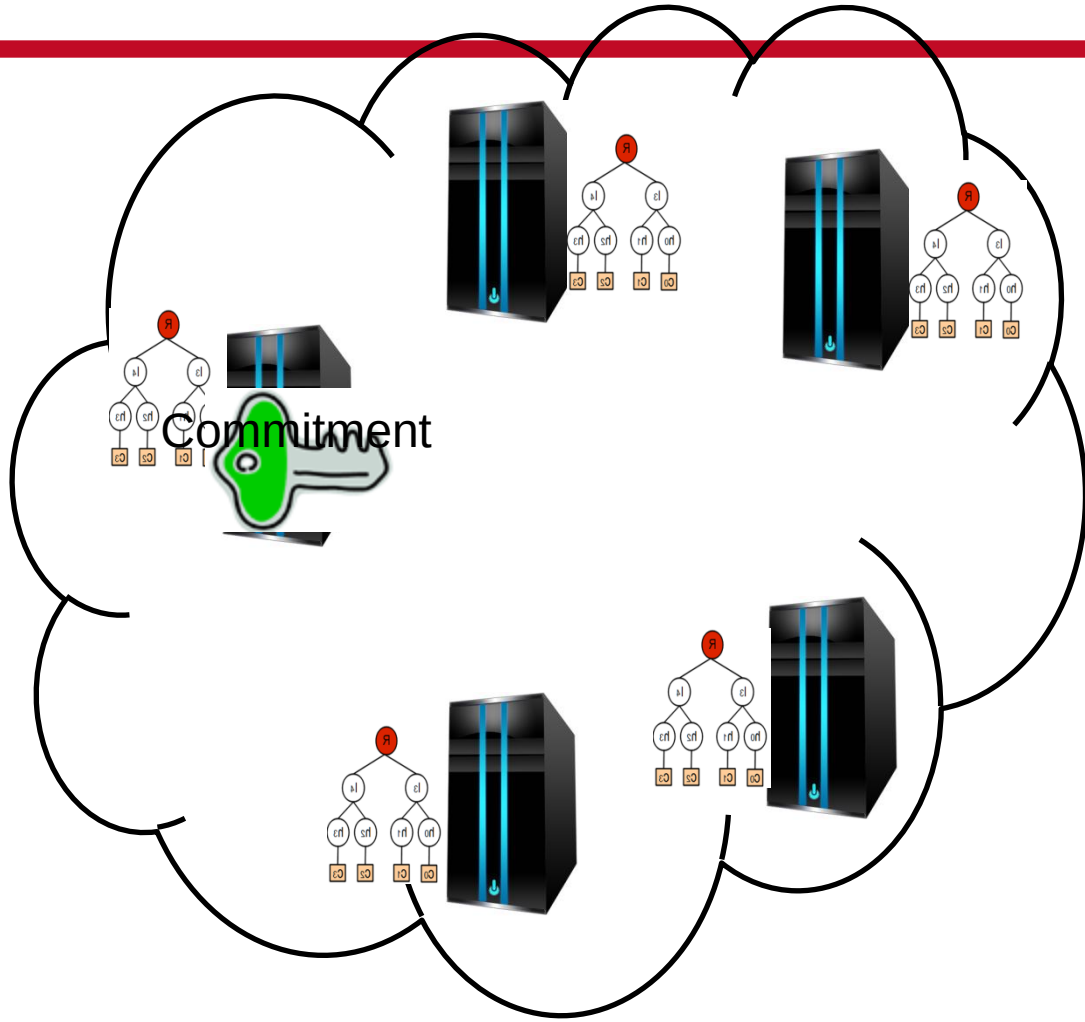
CONIKS

My key

Verify

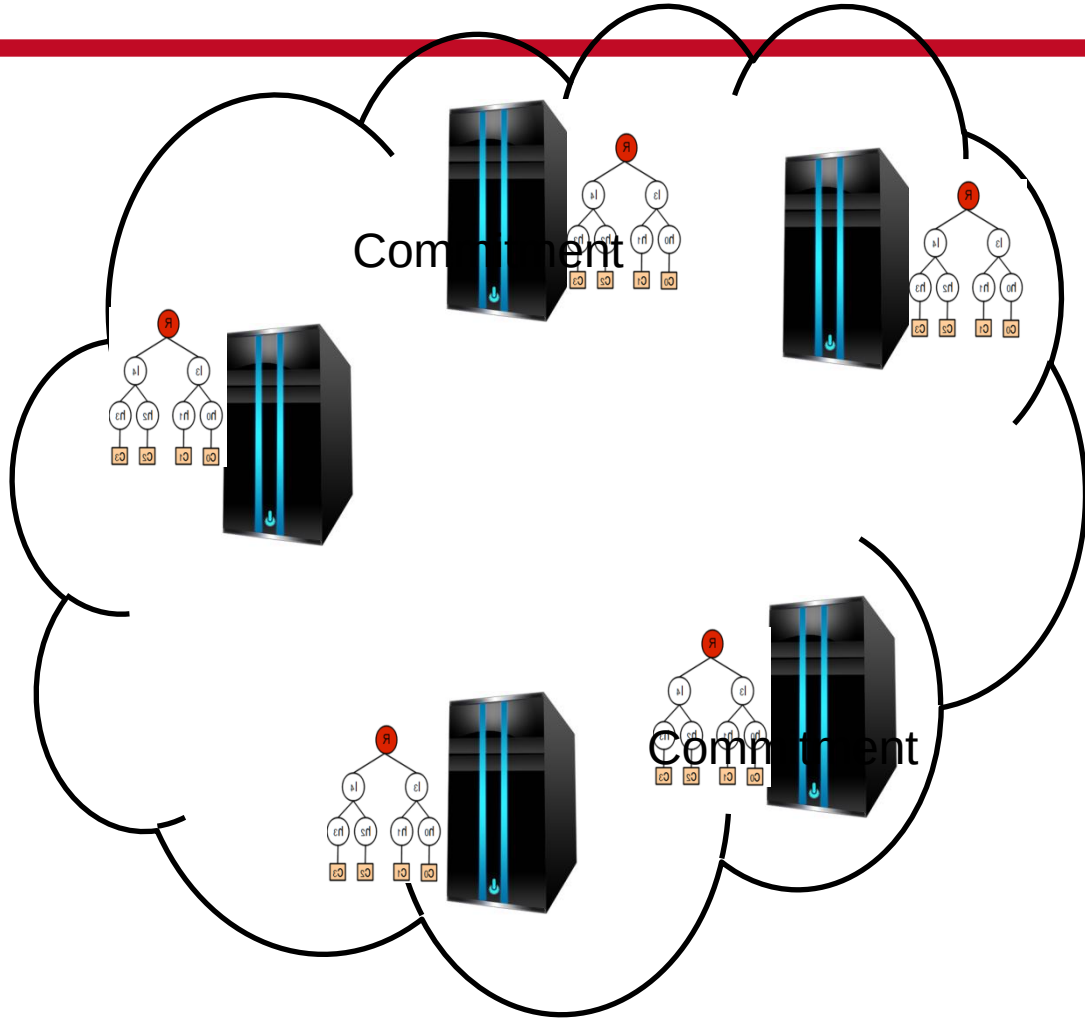


CONIKS



Bob’s key

CONIKS



Problems in Existing Solution

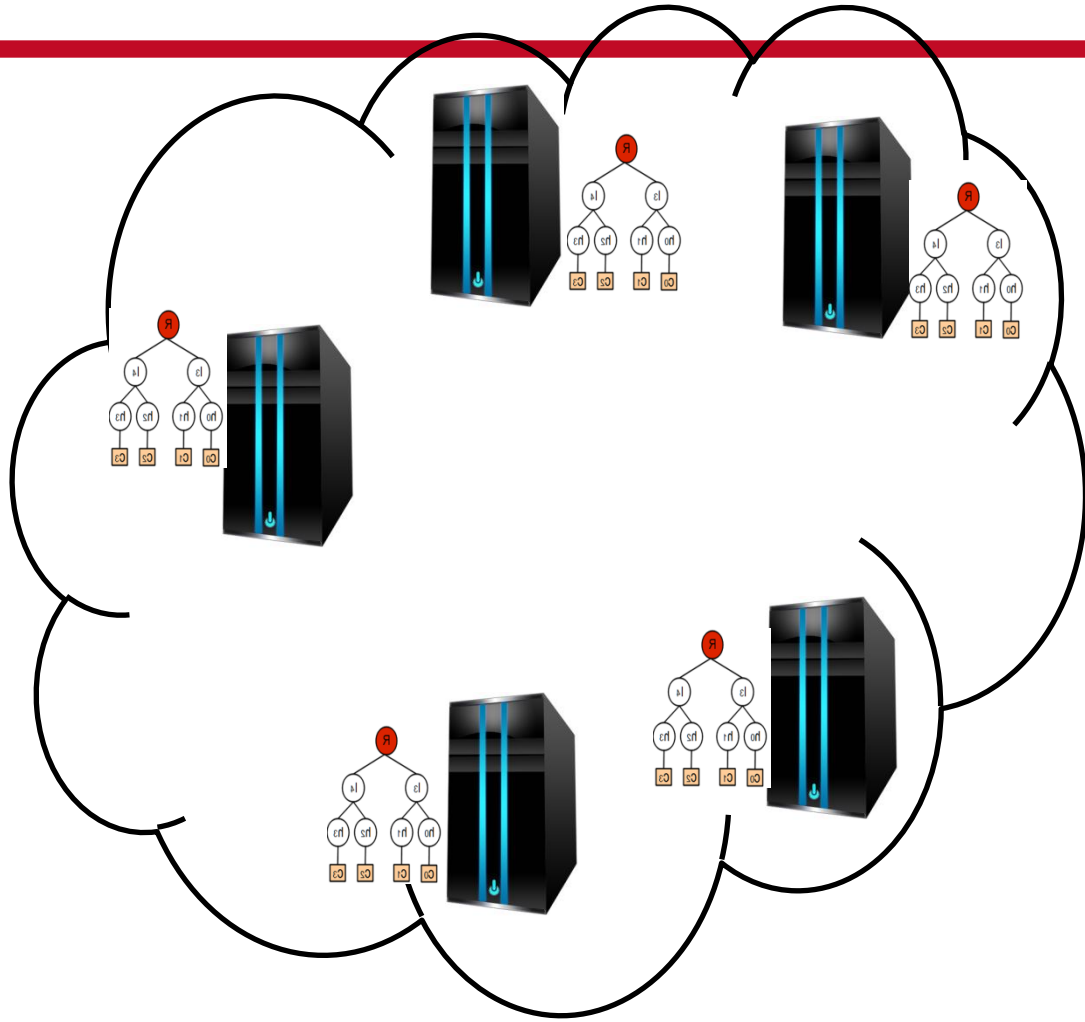
- Organizing the keys in a tree data structure ties them altogether.
  - Even if only one key changes, all users need to check the resulting new tree to make sure they are not affected.
    - Large communications and computations

- **Our Solution:**
- We store the user keys separately
  - Decreasing provider and client computation while increasing the privacy-preservation level.

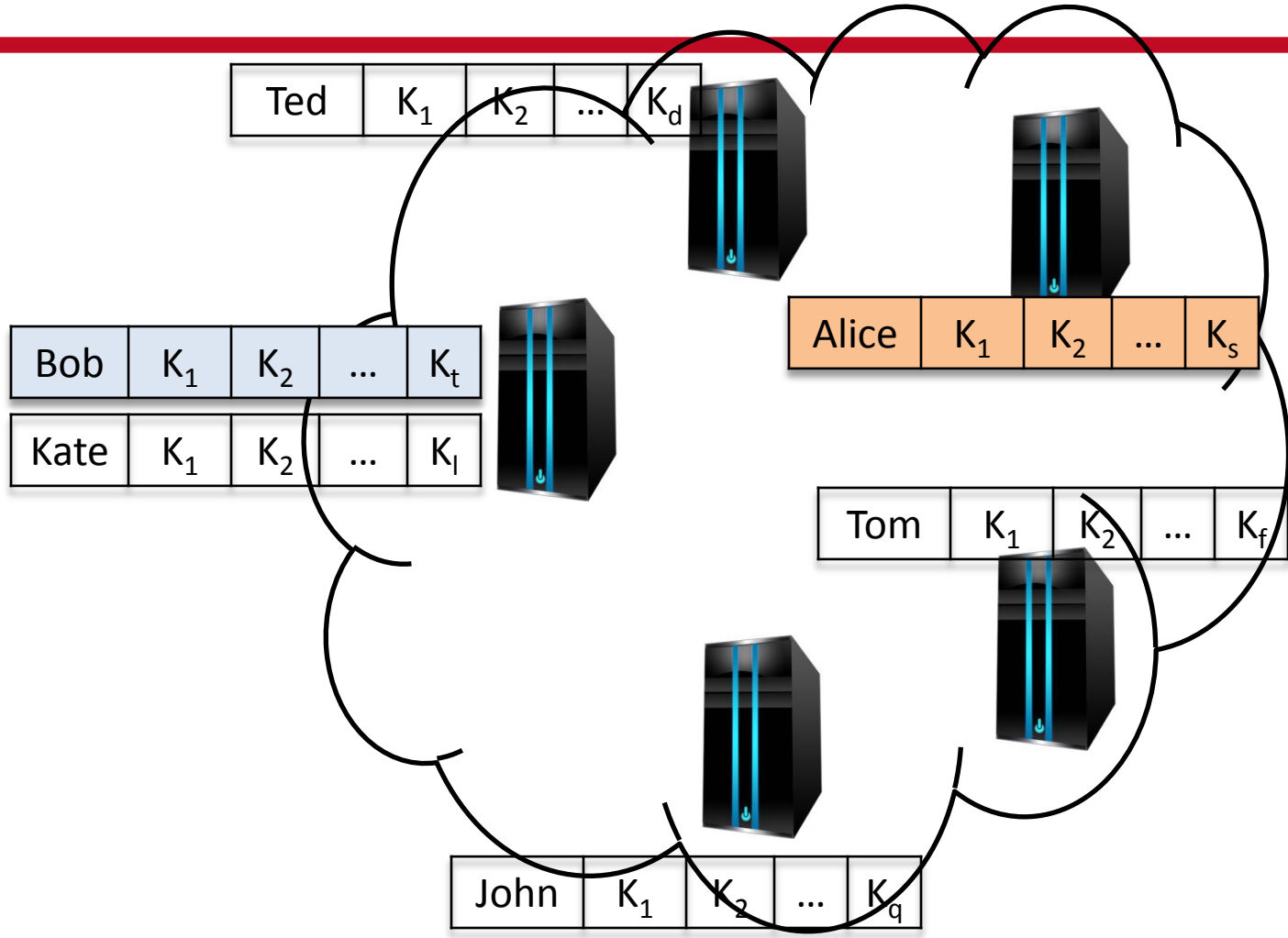
Comparison to Previous Work

Scheme	Provider		User		Gossiping
	Key Reg.	Proof Gen.	Comp.	Proof Size	
Laurie *et al.* [6]	$O(\log n)$	$O(n \log n)$	$O(\log n)$	$O(\log n)$	Client-side
ECT [12]	$O(\log n)$	$O(n \log n)$	$O(\log n)$	$O(\log n)$	Client-side
CONIKS [9]	$O(\log n)$	$O(n \log n)$	$O(\log n)$	$O(\log n)$	Server-side
Our KAS	$O(1)$	$O(1)$ ($O(n)$ audits)	$O(1)$	$O(1)$	Server-side

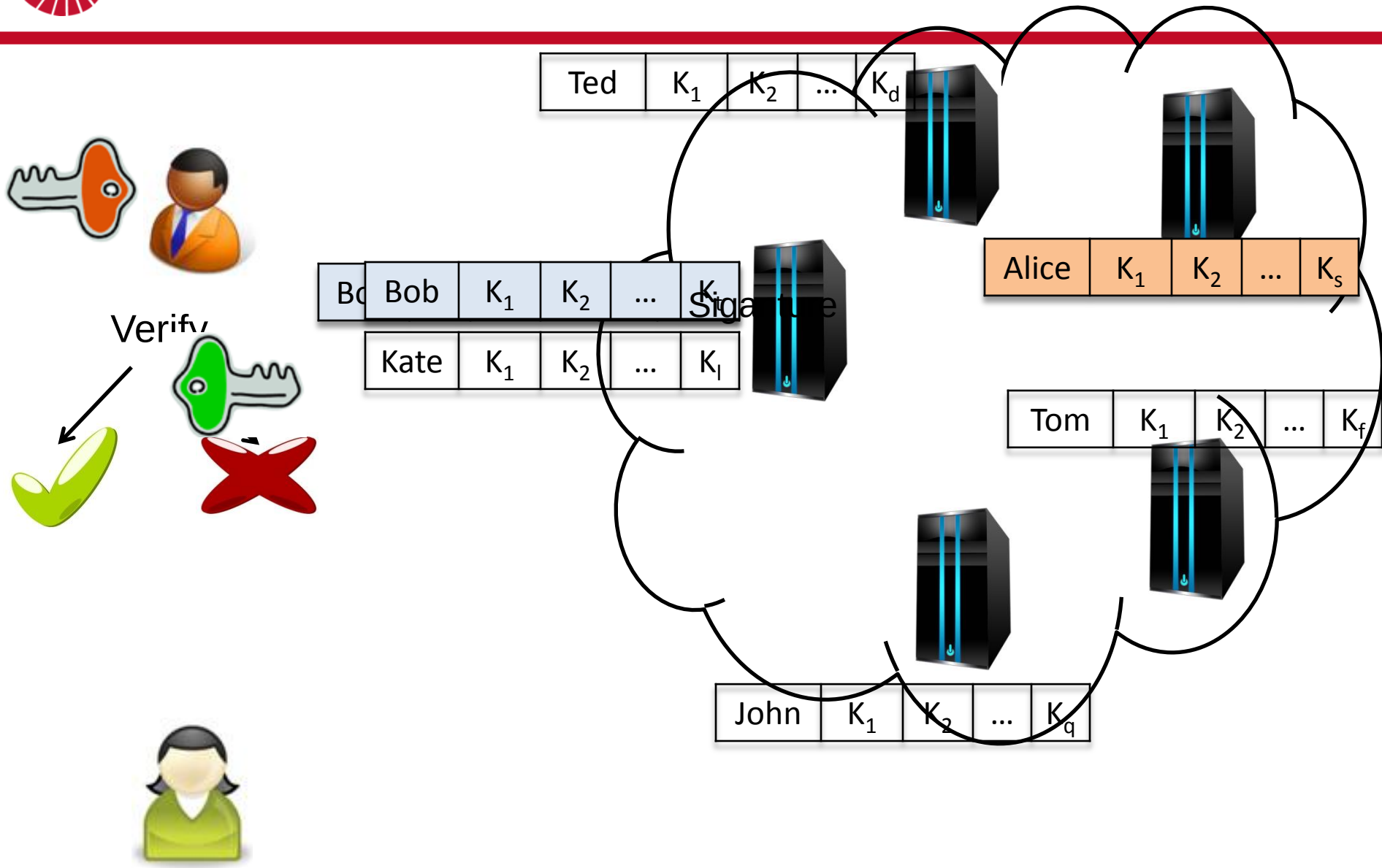
Previous Work



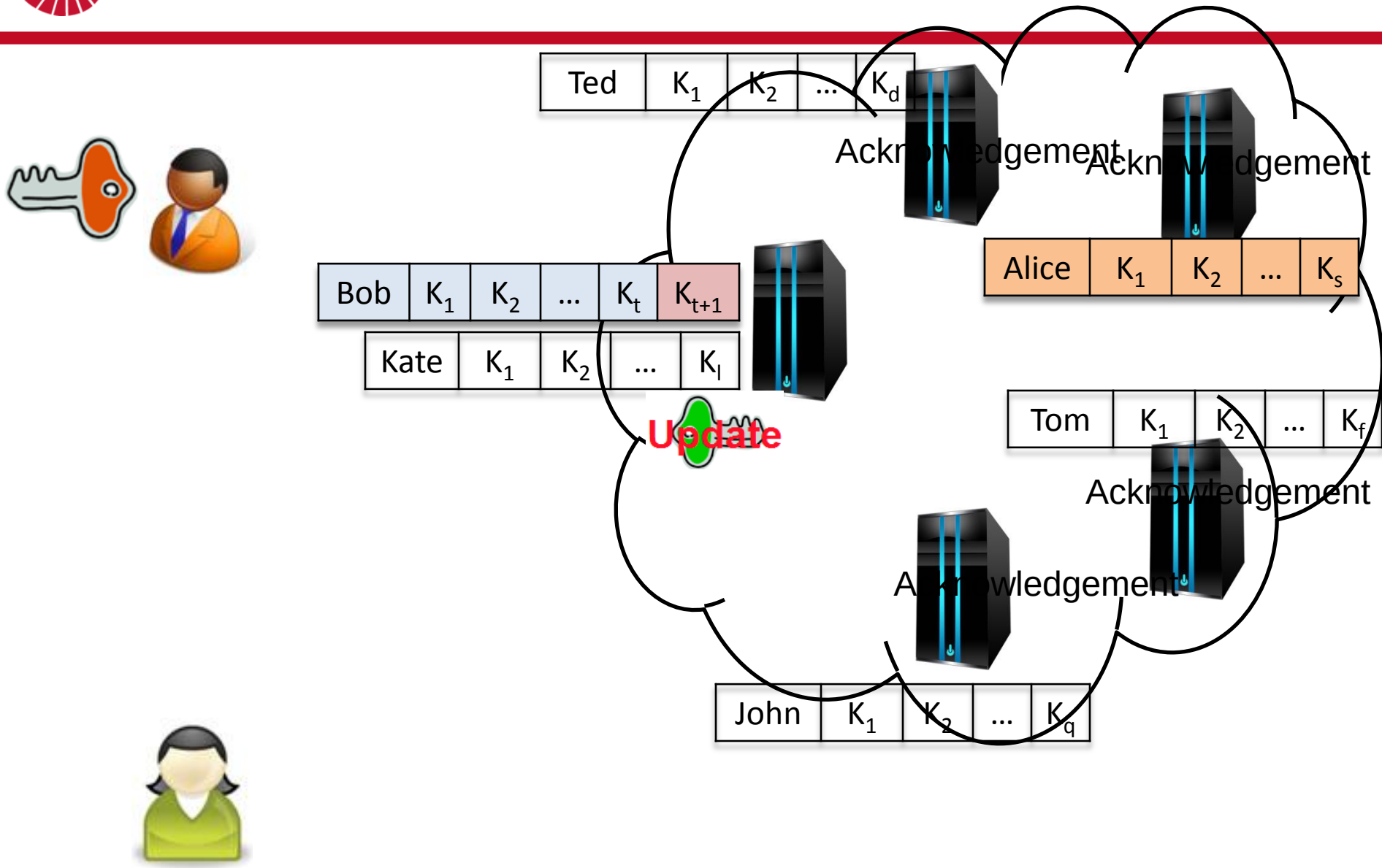
Our Solution



Our Solution



Our Solution



Our Solution

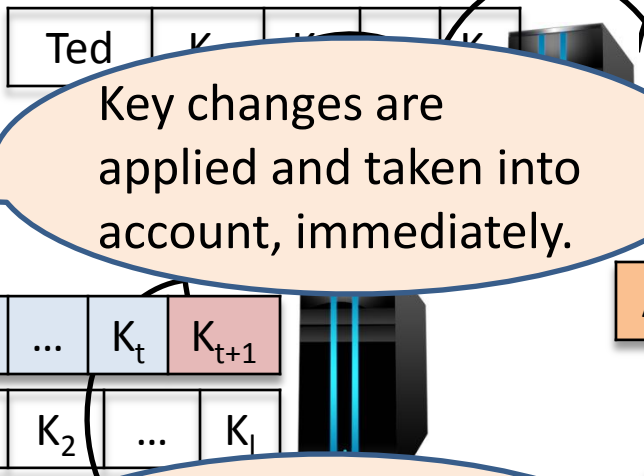
My key

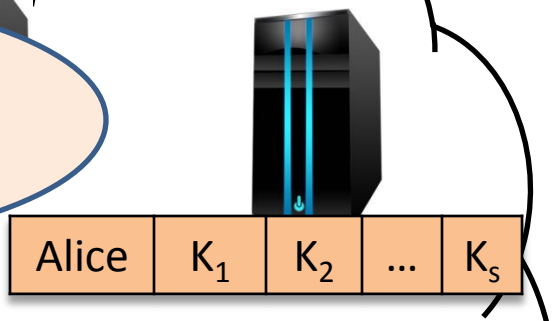
Verify

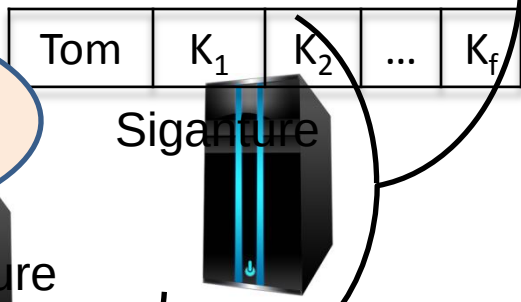
Key changes are applied and taken into account, immediately.

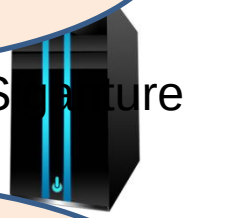
No other certificate owner is affected by Bob's key update.

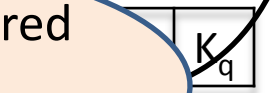
They are not required to do any extra consistency check.











Properties of Our Scheme

- **Privacy-preserving**.
  - Auditing or requesting a user key reveals nothing about the other users.
- **No need for a consistency proof.**
  - On each update, other users should check they have not been affected.
  - No consistency check in ours as the users' data are stored separately.
- **Proof of absence**. On a key request: home provider returns the registered key and his signature (proof of presence).
  - If there was no registered key, our scheme returns $\perp$, as the proof of absence. This is a result of equivocation detection.
- **Non-repudiation**.
  - A common problem is to find the origin of any potential inconsistency.
  - In our scheme, each party commits to all her work or acknowledges others', and stores the related commitments or acknowledgments.
  - No party can later deny his work.

Equivocation Detection

- **Setup**
  - Alice is the key owner that knows her (latest) key and signature.
  - Bob is another user who wants to obtain Alice's key.
  - There are 1000 providers, of which
  - *k* are selected randomly and challenged each time.
  - *e* portion of providers are equivocating:
    - they give the correct signatures to Alice while giving fake signatures (about Alice) to other users and *f* portion of other providers.

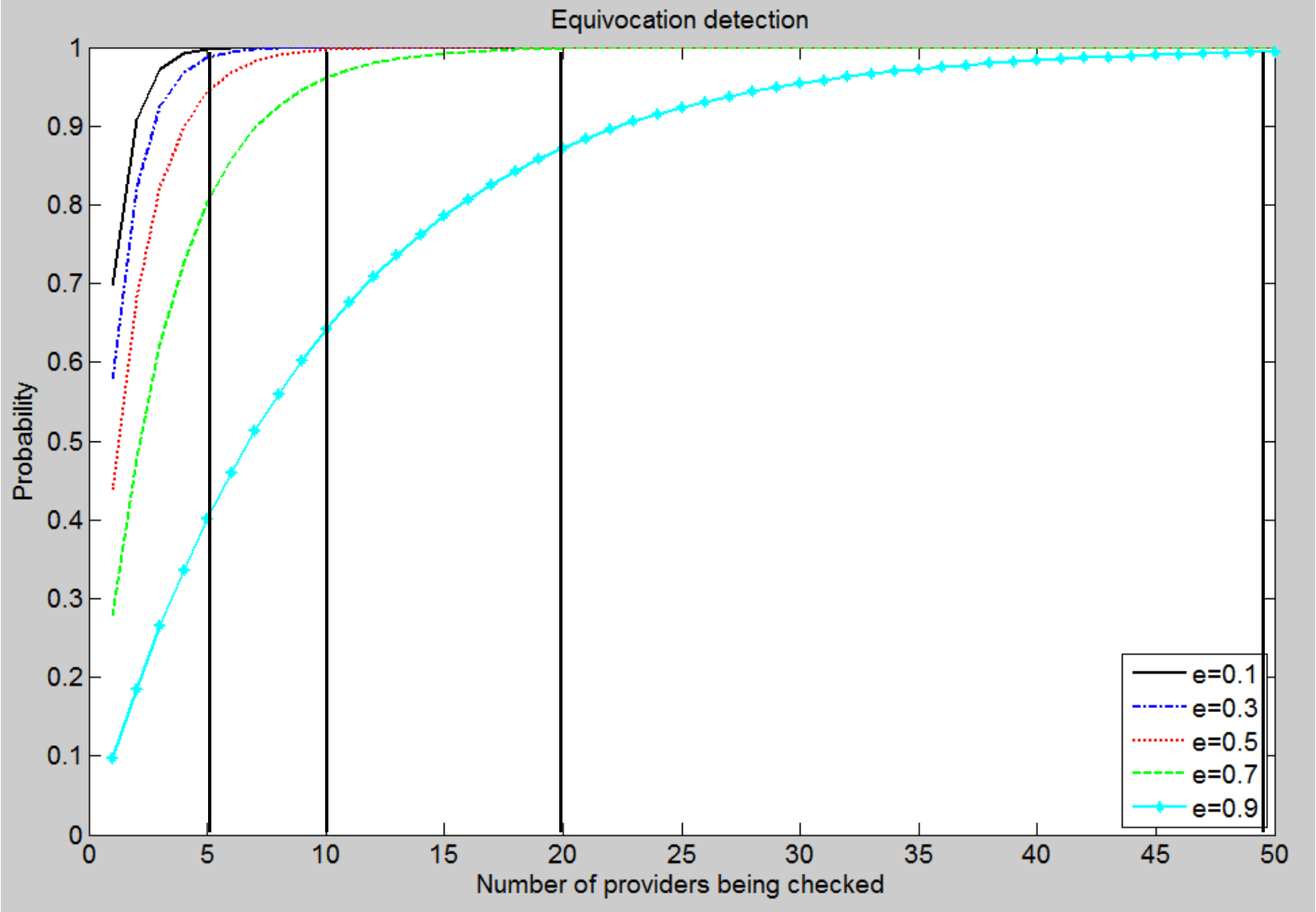
Equivocation Detection

- An equivocation occurs if
  - the key owner receives and accepts the correct key, and
  - another user receives and accepts a fake key.

  - This means that the providers successfully gave a fake key for Alice to Bob while Alice is regularly checking her key.

- Alice accepts the obtained (correct) key with probability $(1-f)^k$.
- Bob will accept a fake key with probability f^k.
- An equivocation occurs with probability $f^k(1-f)^k$.
- The probability of detection is $1- f^k(1-f)^k$.

Equivocation Detection



Performance Analysis

- Scenario:
  - Each provider has n=10M registered users.
  - A user changes her key once a year, *i.e.,* ~ 27,400 changes per day.
  - The security parameter λ = 128.
  - We use SHA-256 as the hash function, and
  - The DSA signature scheme with key pair size (2048,256) bits.
  - The numbers are averages of 50 runs.

- Audit proof size comparison:

Scheme	Provider			User		
	Complexity	One epoch	Per day	Complexity	One Audit	Per day
CONIKS	$O(n \log n)$	915 MB	257.5 GB	$O(1)$	4.68 KB	1.31 MB
Our KAS	$O(n)$	305 MB	85 GB	$O(1)$	1.56 KB	450 KB

Conclusion

- The existing certificate transparency solutions store the keys in a tree data structure, which ties them altogether:
  - extra consistency check
  - Large communications and computations

- We store the user keys separately and achieve
  - Optimal key registration and audit time, and proof size
  - Provide the privacy-preservingness
  - Provide non-repudiation

- We give the first formal security definition of certificate transparency and prove our system security formally.

References

- [LLK13] B. Laurie, A. Langley, and E. Kasper. Rfc 6962: **Certificate transparency**, 2013.
- [LK14] B. Laurie and E. Kasper. **Revocation transparency**. http://www.links.org/files/RevocationTransparency.pdf. Accessed: 20/04/2015.
- [MBF14] M. S. Melara, A. Blankstein, J. Bonneau, M. J. Freedman, and E. W. Felten. **Coniks: A privacy-preserving consistent key service for secure end-to-end communication**, 2014.
- [MR14] M. D. Ryan. **Enhanced certificate transparency and end-to-end encrypted mail**, Proceedings of NDSS, The Internet Society, 2014.
- [NN00] M. Naor and K. Nissim. **Certificate revocation and certificate update**. Selected Areas in Communications, IEEE Journal on, 18(4):561–570, 2000.
- [SGW13] S. G. Weber. **Enabling end-to-end secure communication with anonymous and mobile receivers - an attribute-based messaging approach**. ePrint Archive, Report 2013/478, 2013.
- [WAP08] D. Wendlandt, D. G. Andersen, and A. Perrig. **Perspectives: Improving SSH-style host authentication with multi-path probing**. In USENIX, pages 321–334, 2008.

Thanks for your attention