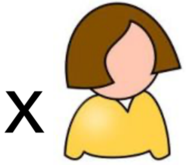
MPCAuth: Multi-factor Authentication for Distributed-trust Systems

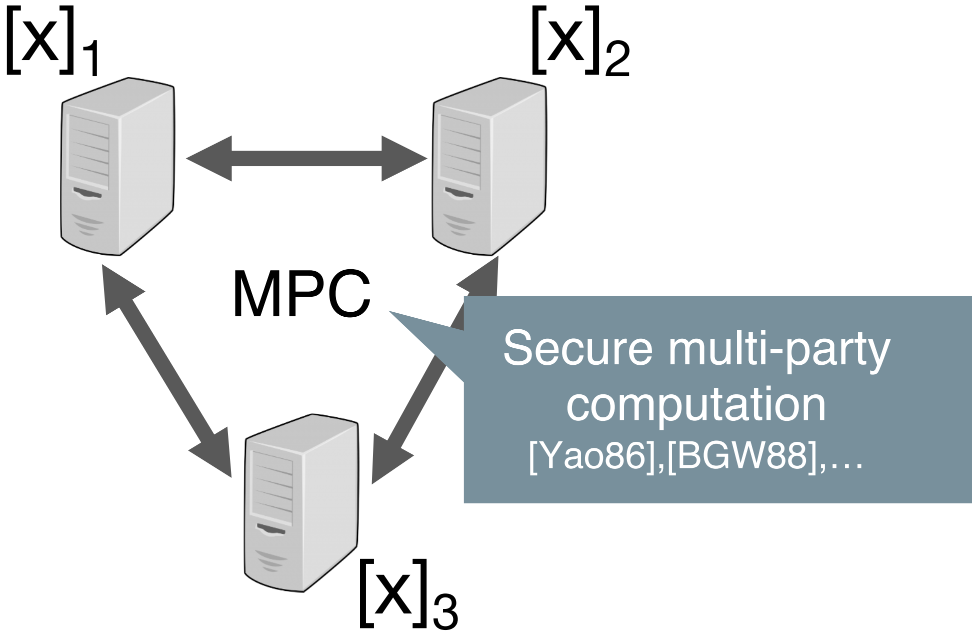
Sijun Tan Weikeng Chen Ryan Deng Raluca Popa

UC Berkeley

IEEE Security & Privacy 2023

Overview of distributed-trust systems

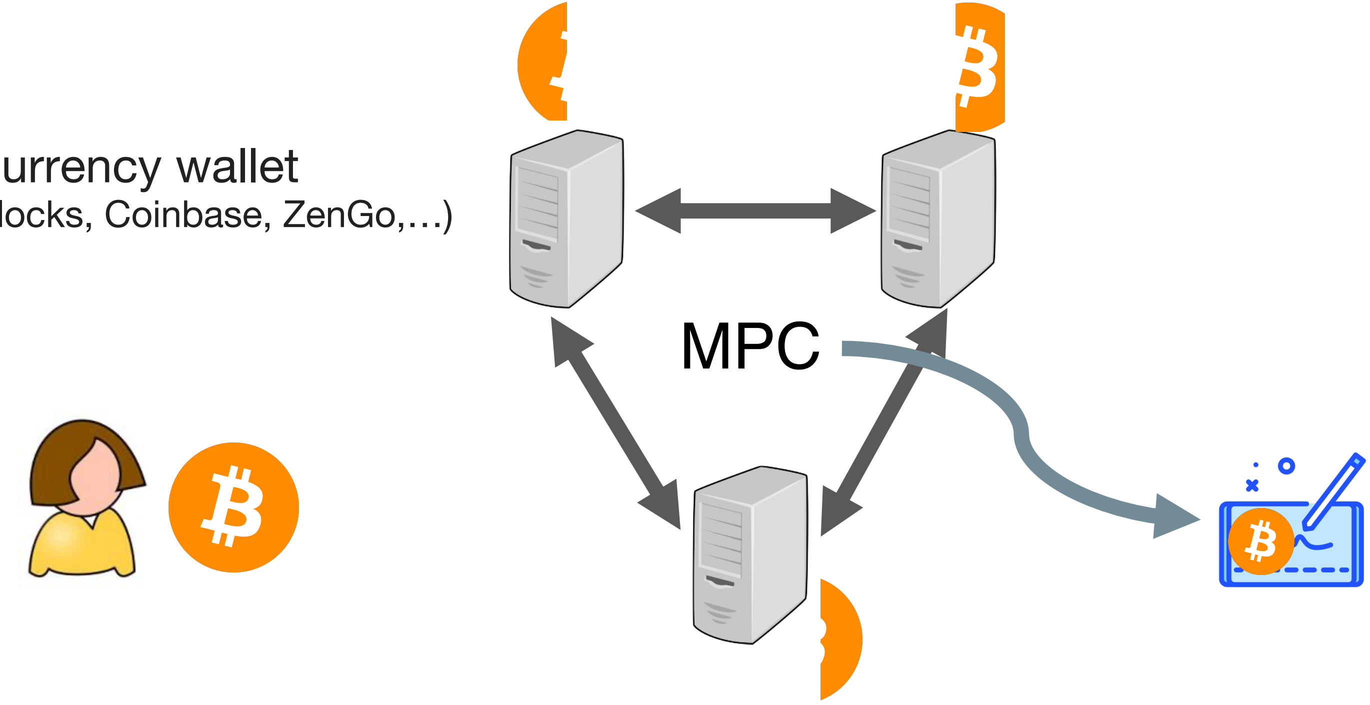




$x = [x]_1 \oplus [x]_2 \oplus [x]_3$

Applications of distributed-trust systems

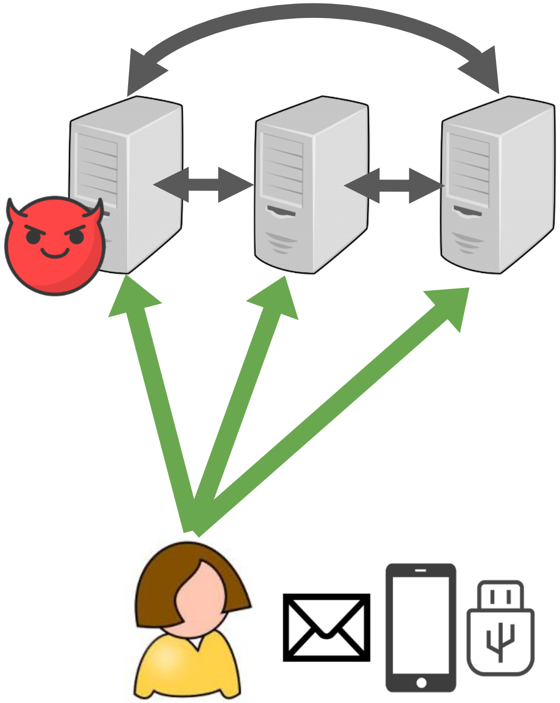
Cryptocurrency wallet
(e.g. Fireblocks, Coinbase, ZenGo,…)



Lots of other applications: Collaborative ML (e.g. Meta, Ant group), Secret key recovery (e.g. Signal) .

How to authenticate to distributed-trust systems?

Strawman 1: Authenticate to one master server.

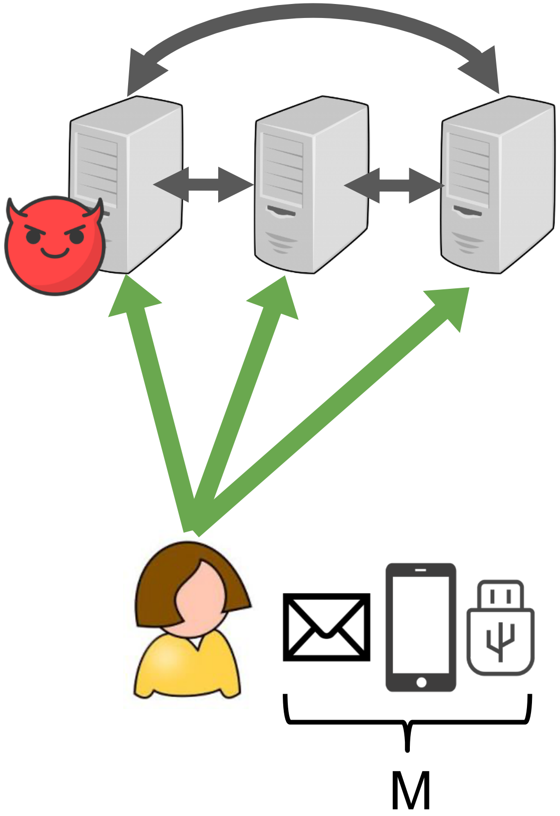


Other servers trust the master server.

A malicious attacker can compromise this one server to recover the secrets.

The client needs to authenticate to all servers to ensure security.

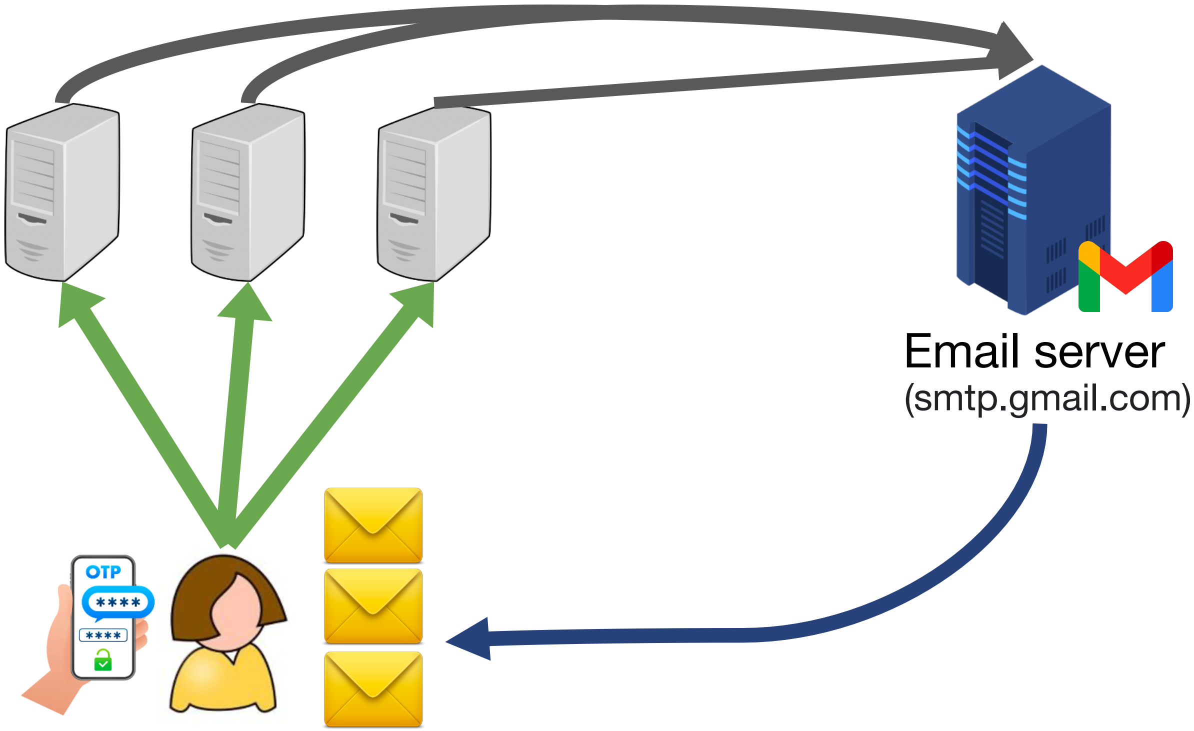
Strawman 2: Authenticate to each of N servers



Avoids a central point of attack.

Problem: The client needs to authenticate to N servers NxM times, one for each of the M factors.

Problem: Burdensome user experience



The client needs to receive N emails and enter passcodes N times!

Our system: MPCAuth

An authentication system for distributed-trust applications in which the user authenticates only ***once***.

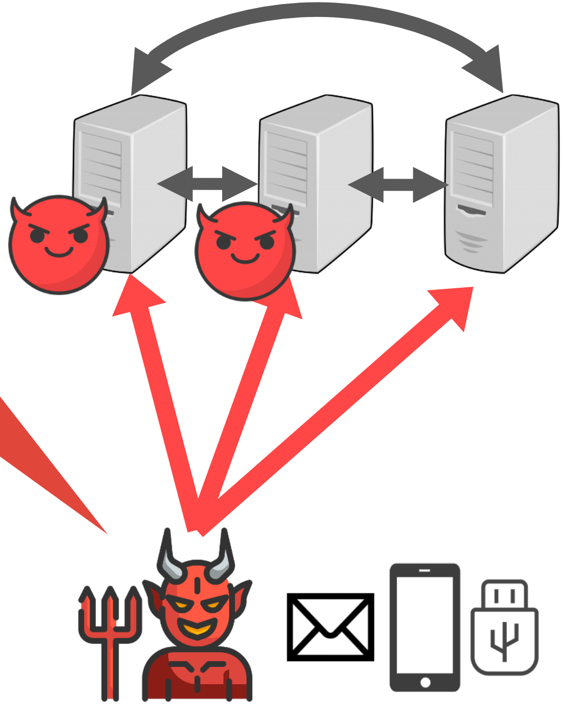
Type	Factors
Possession	Email, SMS, U2F
Knowledge	Passcode, Pin, Security Questions
Inherence	Biometrics

In addition, hides the user's authentication profiles. (e.g. email username, phone number, passwords, biometric features)

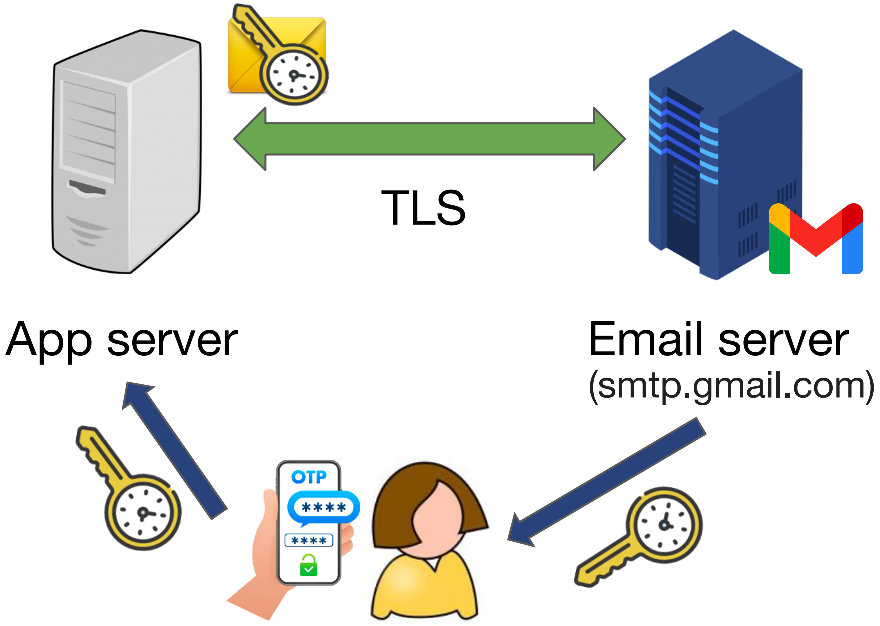
Threat model

- An attacker can corrupt up to N-1 out of N servers.
- The attacker tries to impersonate a client.

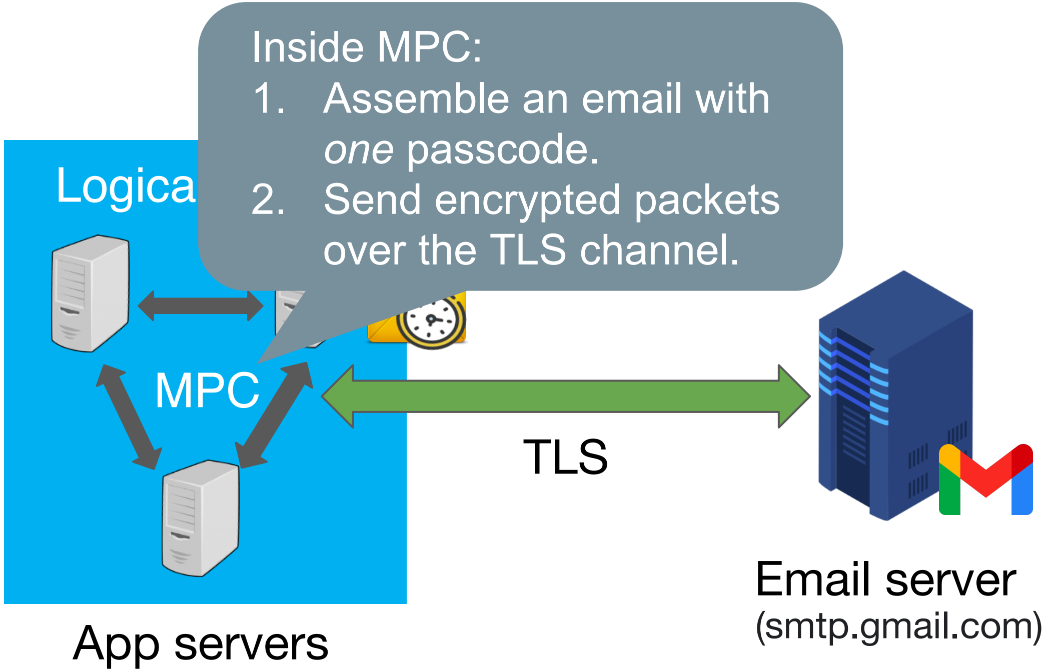
The attacker cannot successfully authenticate as an honest user, if at least one server and one authentication factor is not compromised.



Traditional email authentication

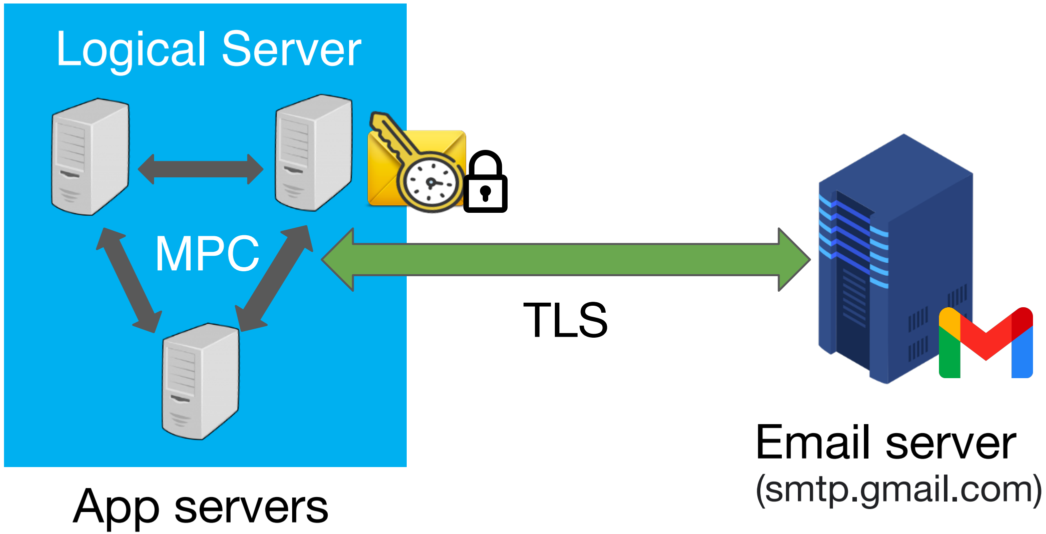


Email authentication for distributed-trust systems



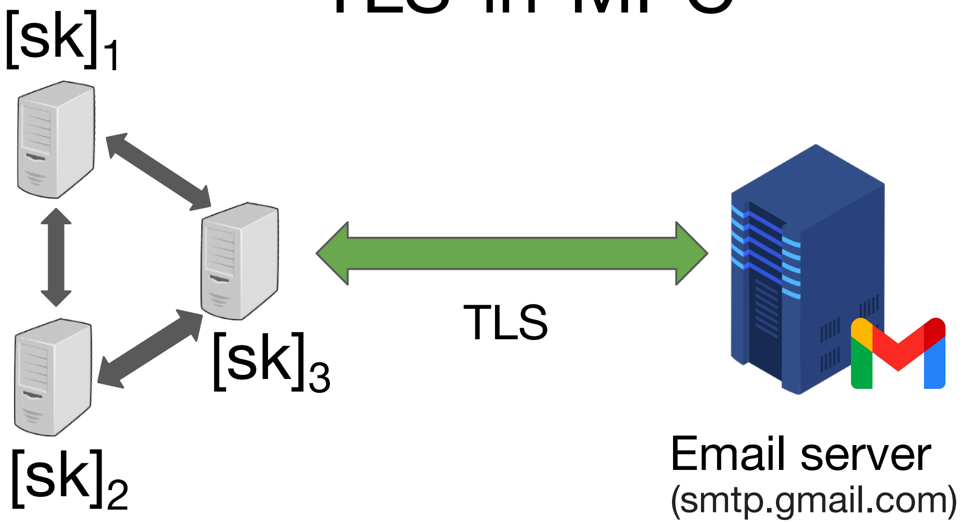
The N servers jointly act as one logical server to interact with the email server.

Email authentication for distributed-trust systems



The N servers jointly act as one logical server to interact with the email server.

TLS-in-MPC

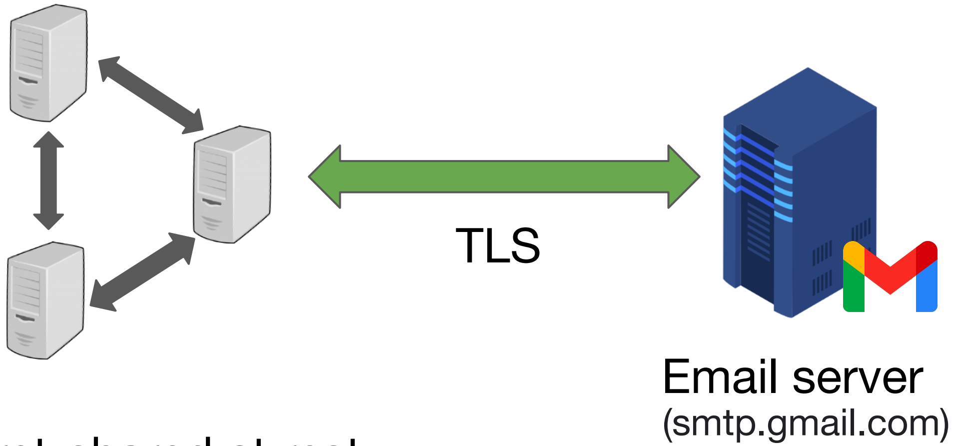


enc,mac:=AES-GCM([sk], [msg])

TLS Handshake: Jointly perform Diffie-Hellman key exchange.

Data transmission: Jointly run an authenticated encryption scheme to encrypt messages and transmit them over the network.

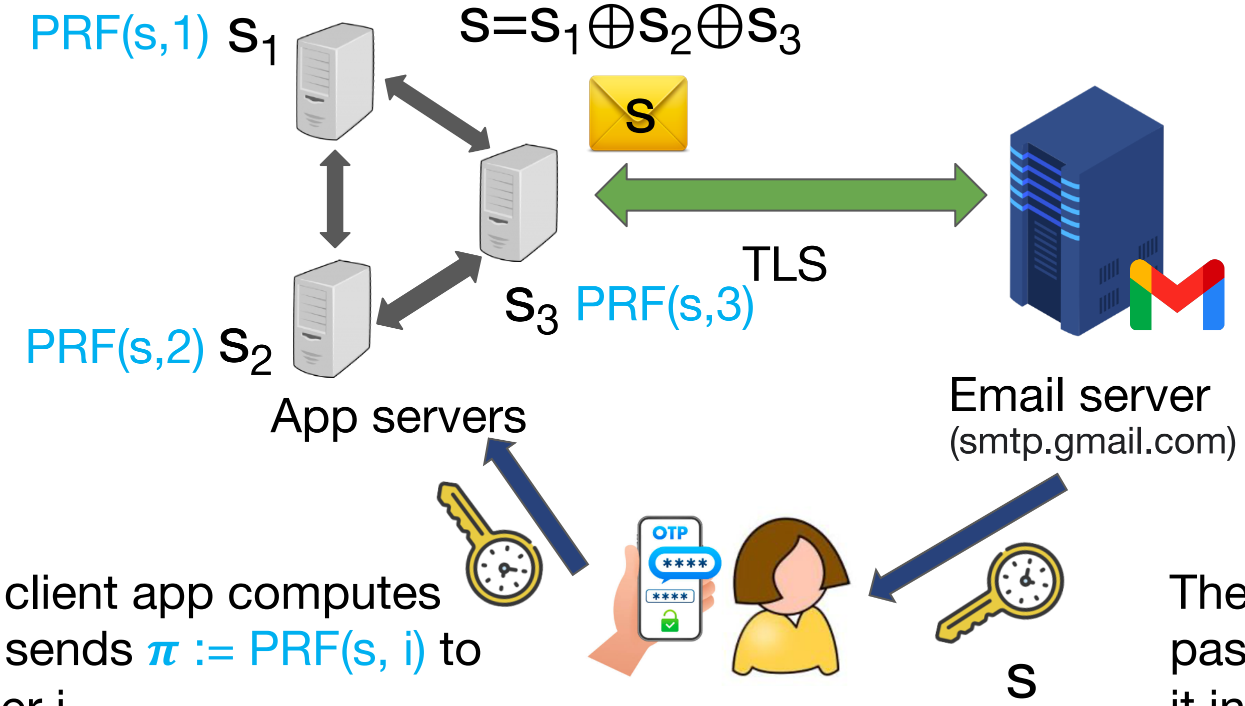
Implication of TLS-in-MPC



- Data is secret-shared at rest.
- During transmission, data is encrypted in MPC with a secret-shared encryption key.
- None of the server sees any plaintext data during the whole process.

The protocol itself is extendable to use cases beyond authentication.

MPCAuth's email authentication protocol

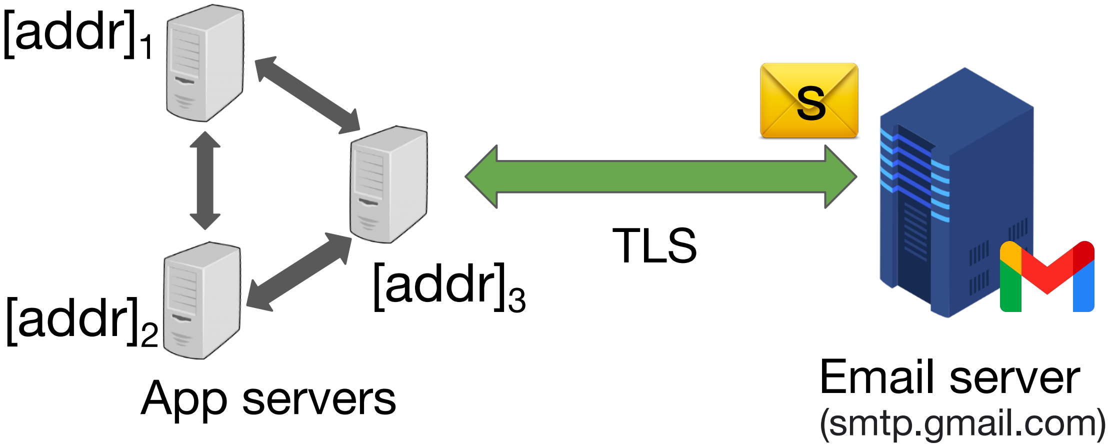


The client app computes and sends π := PRF(s, i) to server i

The client receives passcode s and enters it into the app.

The passcode s is hidden from all servers.

MPCAuth's email authentication protocol



- The client only enters the passcode *once* on the client app.
- The client's email username is hidden from all servers.

Implementation & Evaluation

Implemented the system using MP-SPDZ, EMP-AGMPC, and WolfSSL.

Evaluated the system on 2-5 AWS c5n.2xlarge 3.0GHz 8 core CPU.

Server-to-server bandwidth: 2Gbit/s
Client-to-server bandwidth: 100Mbit/s.

Without established TLS

3PC	**Offline**	**Online**	**Total**
Email Auth	10.9s	1.3s	12.2s

With established TLS

3PC	**Offline**	**Online**	**Total**
Email Auth	2.9s	0.4s	3.3s

Works with existing email provider (Gmail) with no timeout.

Summary of MPCAuth

An authentication system for distributed trust applications.

- Enables a client to authenticate independently to N servers by doing the work of only *one* authentication.
- Design secure, practical, and profile-hiding protocols for multiple authentication factors.

Email: sijuntan@berkeley.edu

Paper: https://eprint.iacr.org/2021/342.pdf

Thank you!