

Ticket to Hide:

Practical, Private Proofs of Provenance for TLS Traffic

Ryan Little

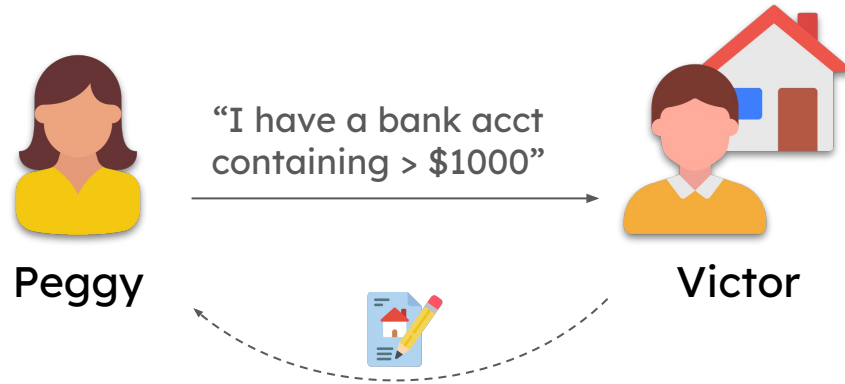
Dan Roche

Mayank Varia



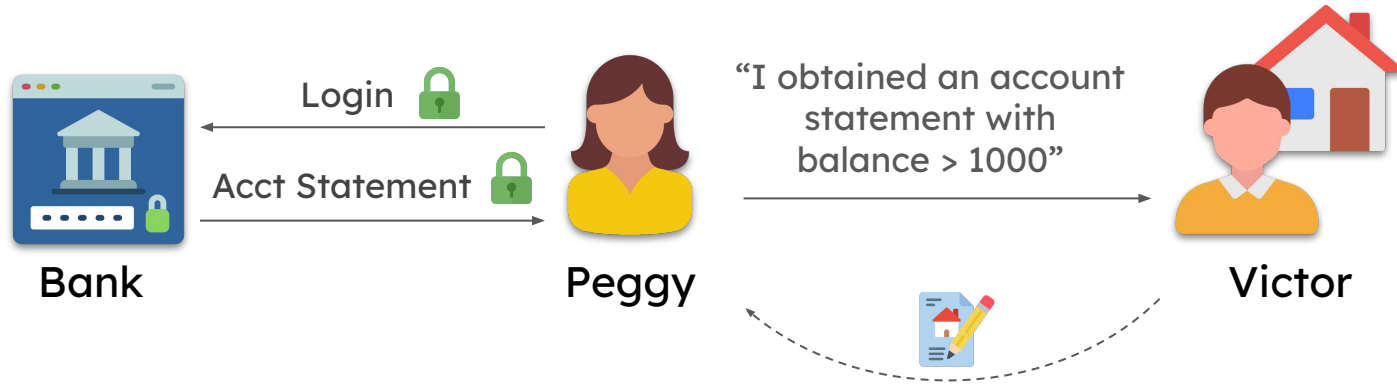
Motivation

- Peggy wants to rent a house from a landlord, Victor
- Victor wants to check Peggy's bank account has $> \$1000$
- Peggy doesn't want to reveal her exact balance, or even which bank she uses



Motivation

- Peggy wants to rent a house from a landlord, Victor
- Victor wants to check Peggy's bank account has $> \$1000$
- Peggy doesn't want to reveal her exact balance, or even which bank she uses



TLS Oracles

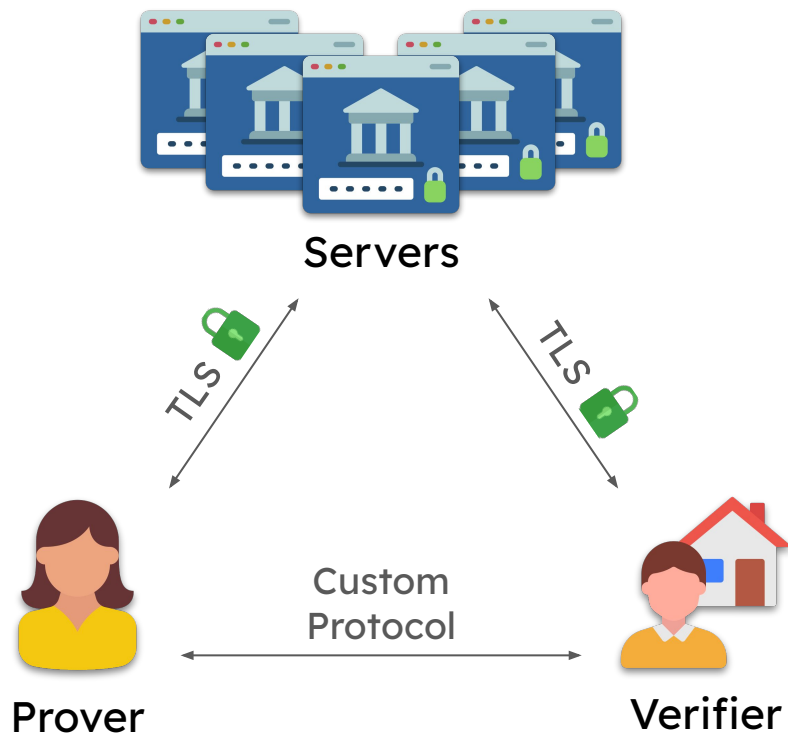
[Zhang Maram Malvai Goldfeder Juels '20]

[Xie Yang Wang Wu '24]

[Lauinger Ernstberger Finkenzeller Steinhorst '25]

[Celi Davidson Haddadi Pestana Rowell '25]

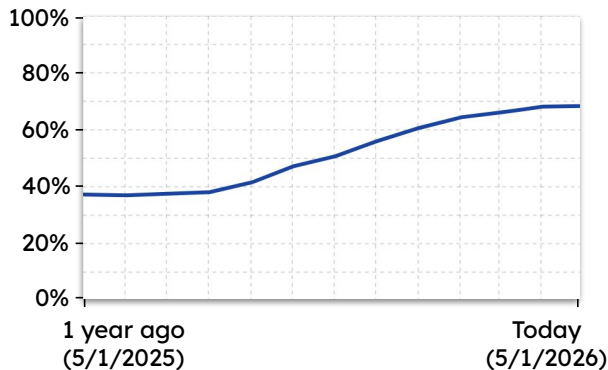
- Prover must convince verifier they received some data from a trusted server, satisfying certain properties
- Set of N publicly known servers, each **only runs TLS**
 - Servers can't run a bespoke protocol!
- Either the prover or verifier can be malicious. A malicious party can control the entire global network.



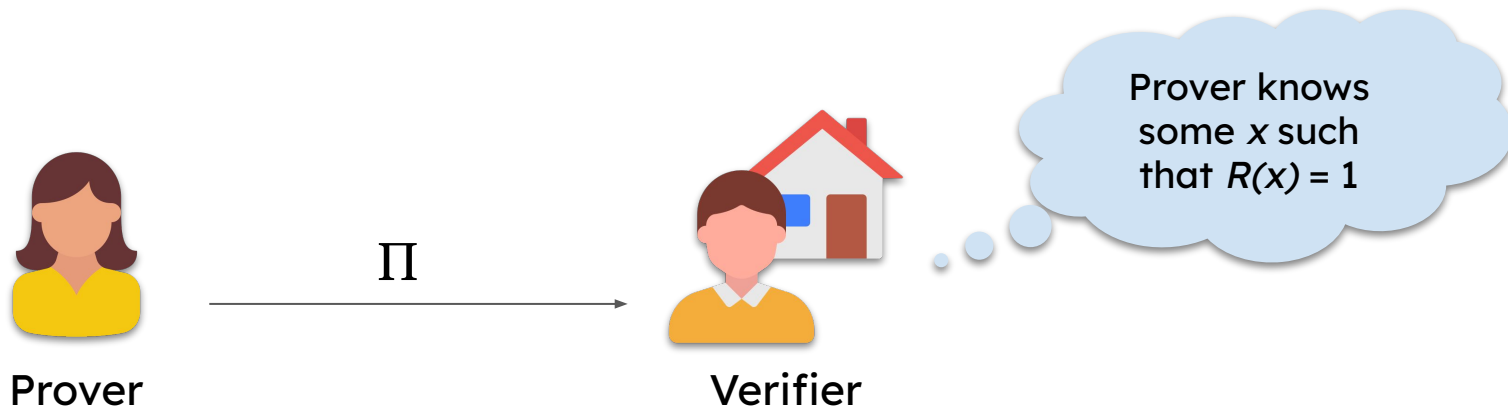
Our Contributions

- **Ticket to Hide:** the fastest TLS Oracle protocol yet!
 - 4-15× faster than state-of-the-art protocols for TLS 1.3
- We leverage two TLS 1.3 features for efficiency gains: session tickets and encrypted client hello
- We are the first TLS Oracle compatible with post-quantum TLS modes

Support for PQ Key
Exchange among TLS
clients (per Cloudflare)

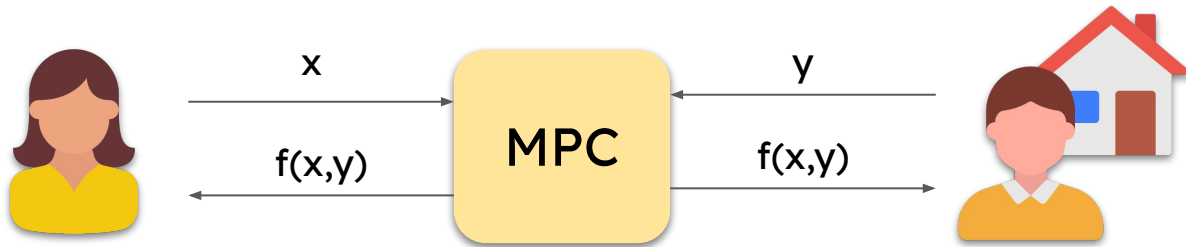


Building Block 1: Zero-Knowledge Proofs of Knowledge (ZKPoKs)



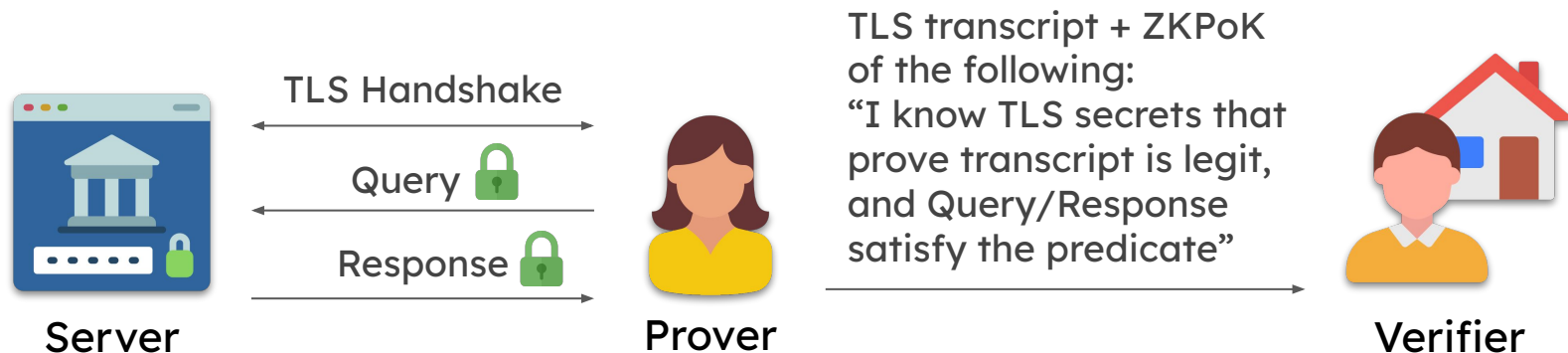
Prover convinces verifier they know a secret that satisfies a predicate,
without revealing anything about the secret

Building Block 2: Secure Multi-Party Computation (MPC)



Multiple parties with secret inputs can compute functions over their secrets, *without revealing their inputs to each other*

A Strawman Protocol



Problem: Prover learns encryption keys during the handshake. She can create a fake transcript by encrypting whatever messages she likes!

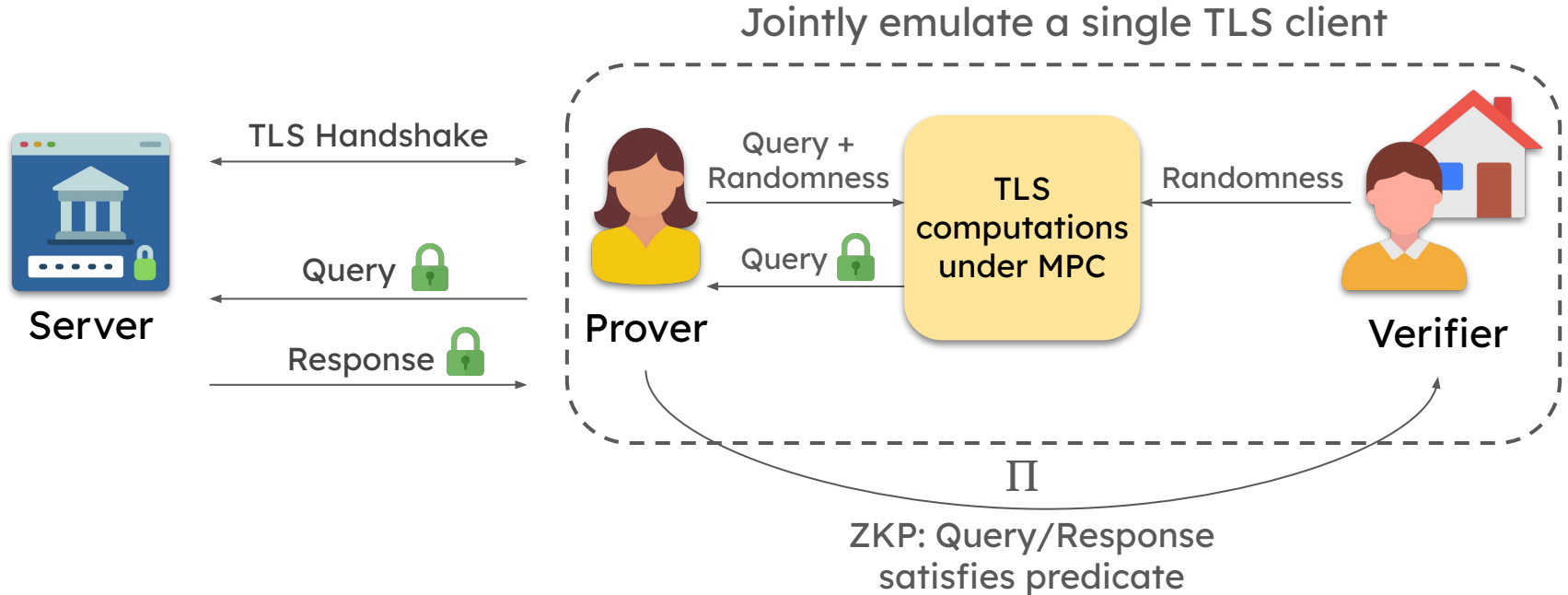
Framework of TLS Oracles

[Zhang Maram Malvai Goldfeder Juels '20]

[Xie Yang Wang Wu '24]

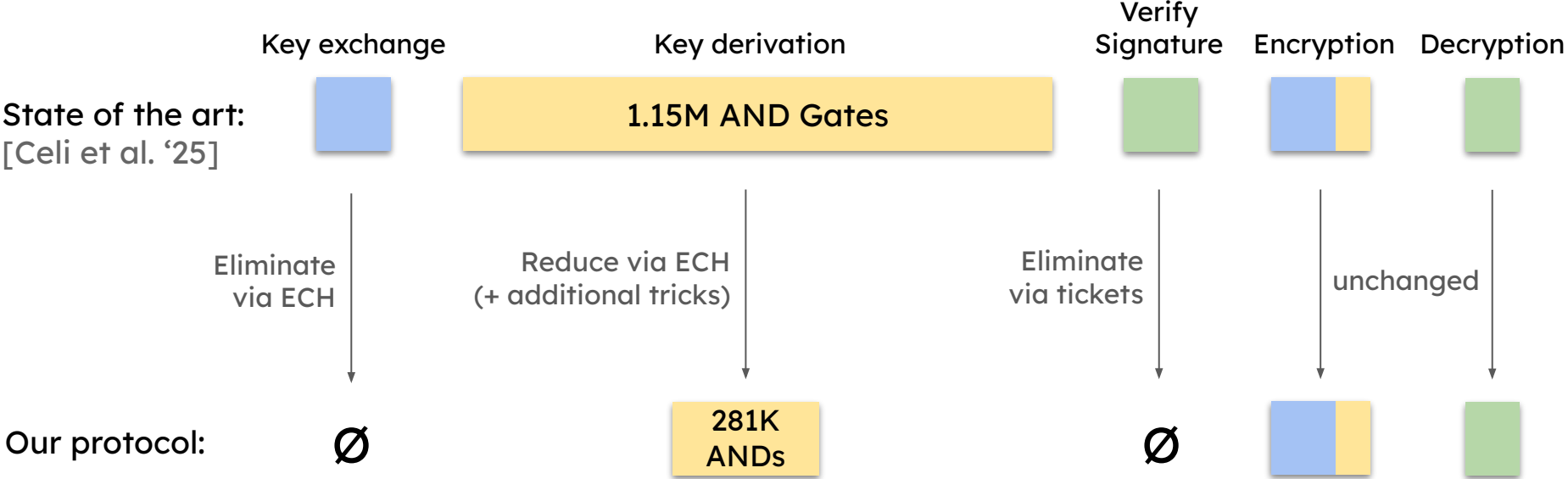
[Lauinger Ernstberger Finkenzeller Steinhorst '25]

[Celi Davidson Haddadi Pestana Rowell '25]



MPC/ZK Comparison

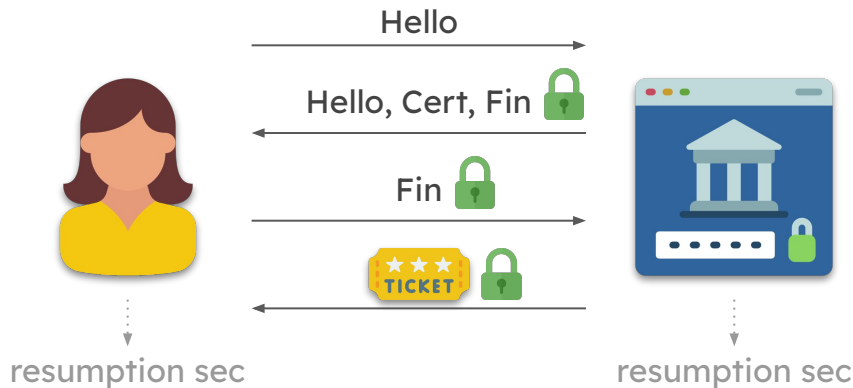
-  = Arithmetic MPC
-  = Boolean MPC
-  = Zero-Knowledge Proof



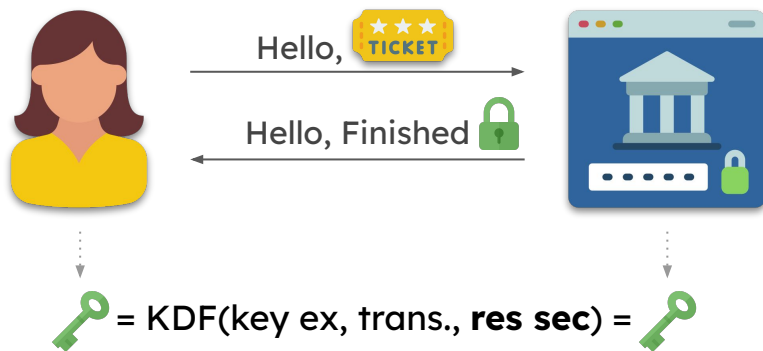
TLS 1.3 Feature 1: Session Resumption Tickets

Client and server can establish a ticket during the handshake that lets them reconnect faster in future sessions

Initial handshake



Reconnection

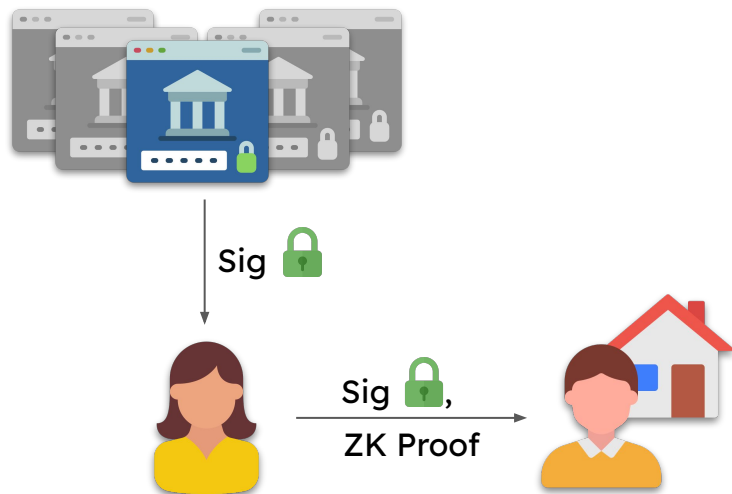


Benefit to us: Client verifies signature in initial handshake, not reconnection

Using Tickets to Prove Provenance to the Verifier

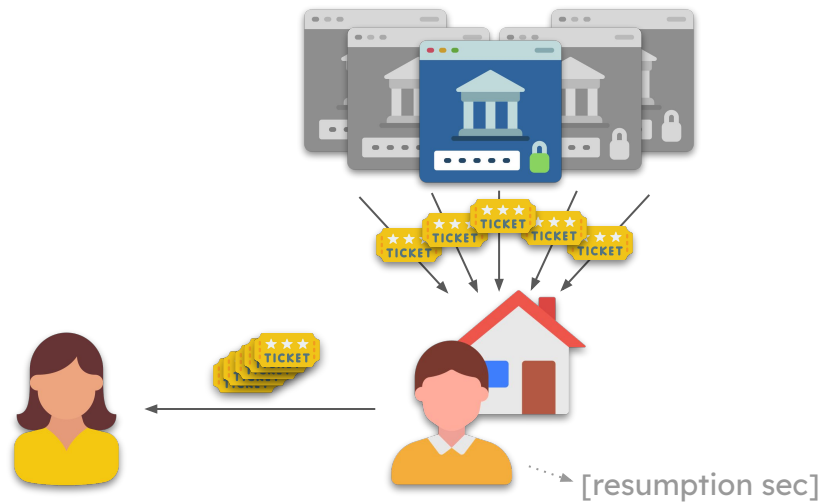
[Celi et al. '25]

- Prover gets signature from chosen server
- Prover gives verifier a ZKP proving they have a signature signed by one of the servers in the set



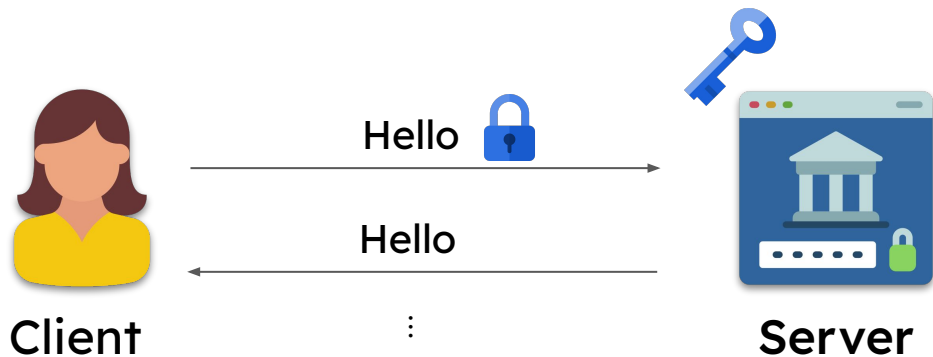
Our Protocol

- Verifier gets a ticket from all servers, checking signatures in the process
- Verifier sends tickets to prover
- Prover and verifier do a single reconnection in 2PC



TLS 1.3 Feature 2: Encrypted Client Hello (ECH)

Clients encrypt Client Hello message under a long-term server public key

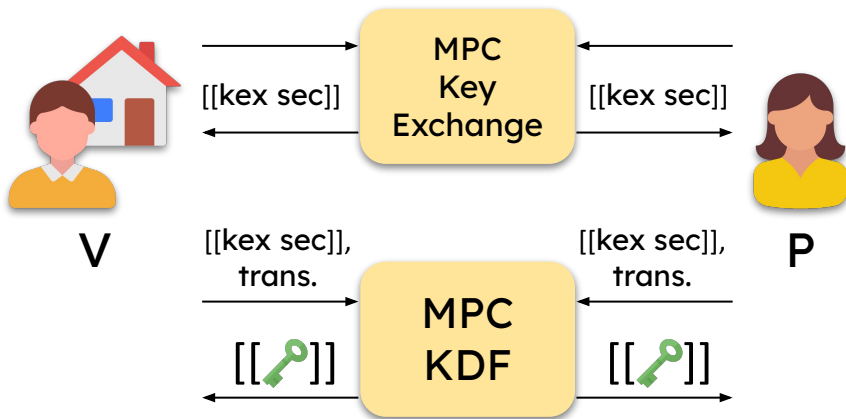


Benefit to us: Client can encrypt randomness in the Hello message that only the server can decrypt

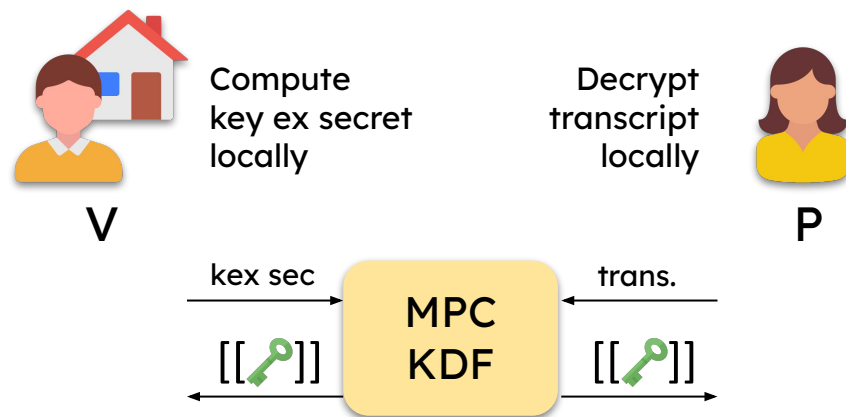
Using ECH to Avoid MPC Key Exchange

We need to compute secret shares of  = $\text{KDF}(\text{key ex secret}, \text{transcript})$

Prior TLS Oracles

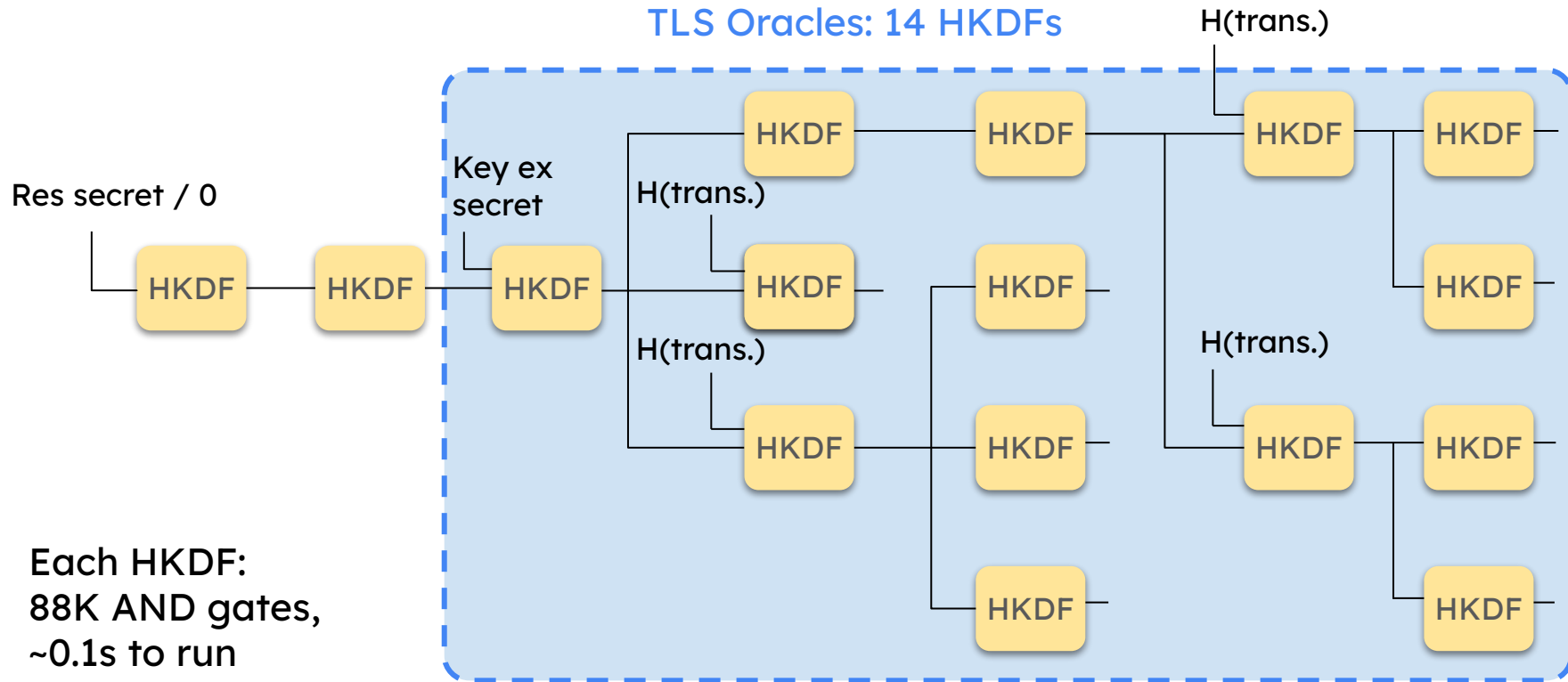


Our Protocol

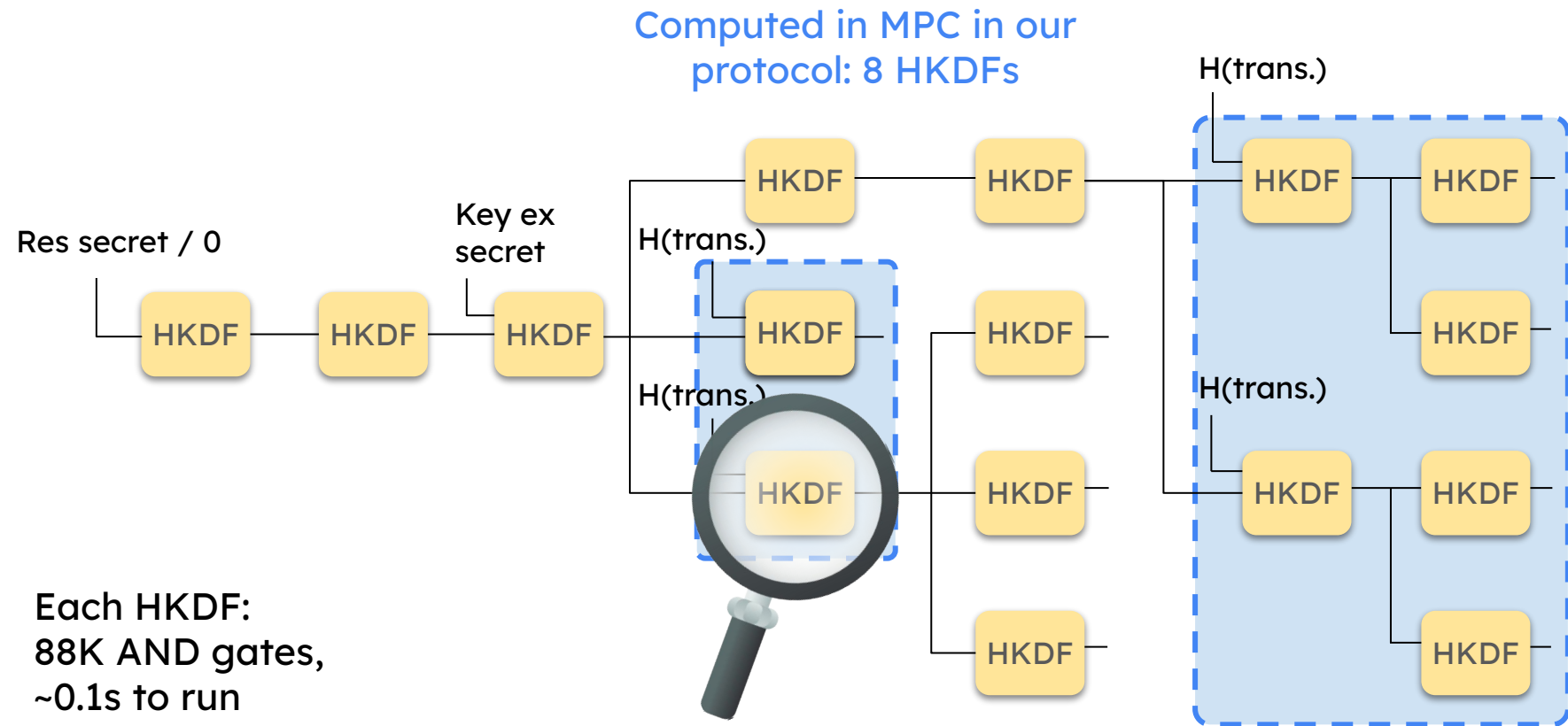


Using ECH to Optimize Key Derivation MPC

Computed in MPC in prior
TLS Oracles: 14 HKDFs



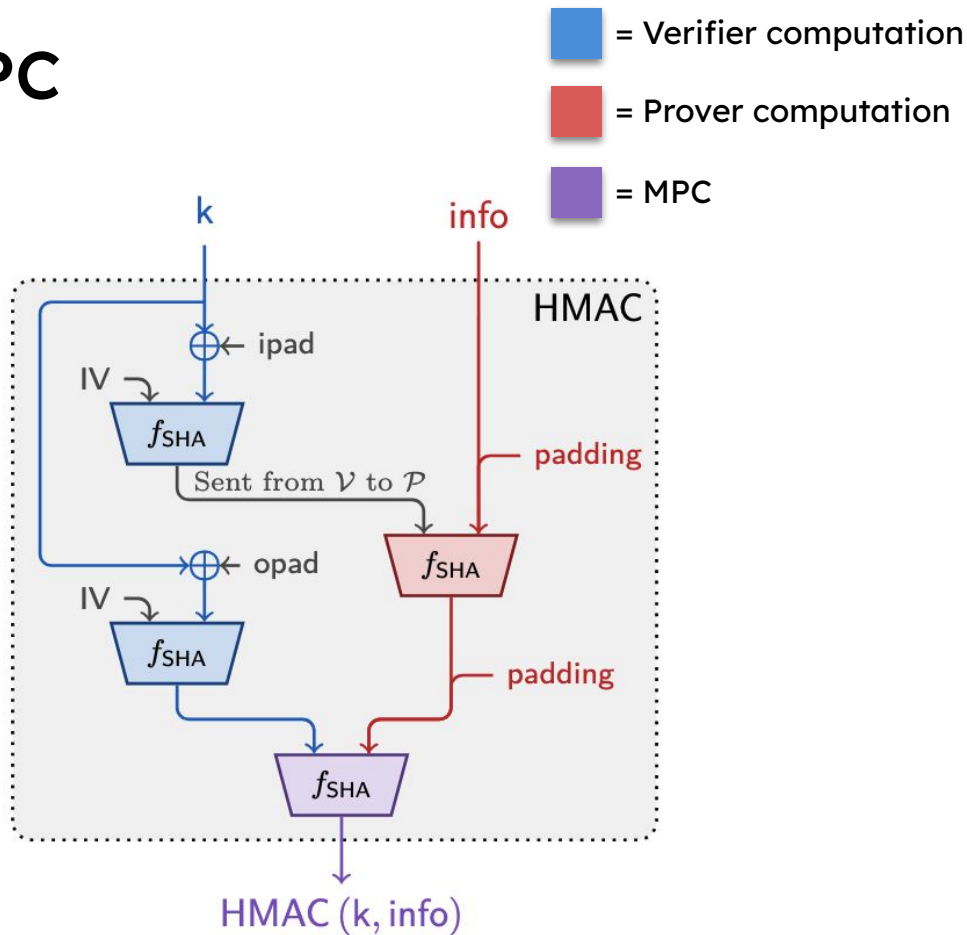
Using ECH to Optimize Key Derivation MPC



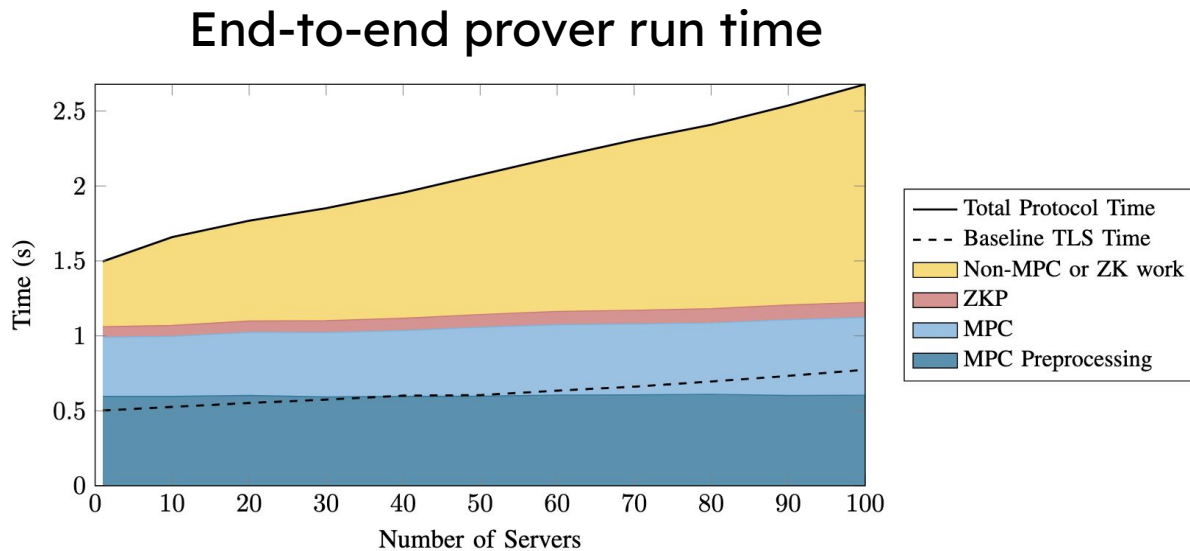
Optimizing HKDFs in MPC

- Each HKDF consists of an HMAC-SHA256 with four SHA-256 compression functions
- With our secret sharing style, **only one** of these needs to be computed under 2PC

ANDs per HKDF: 88K $\rightarrow$ 22K



Evaluation



- Full protocol runs in under 3 seconds over WAN—fast enough to be feasible in practice
- Only 3-4 times the overhead of a *regular* TLS connection, for up to 100 servers

Summary

- Ticket to Hide lets you privately prove statements about data sent via TLS to a designated third party, while also hiding the identity of the server among a set of possible servers.
- We leverage TLS 1.3 features like session resumption tickets and encrypted client hello to get significant efficiency improvements, as well as support for post-quantum TLS.

Thank you!



Full Paper