Oblivious HTTP
draft-thomson-ohai-ohttp
Problem Statement
Background

Linking client \textit{data} with client \textit{identity} is problematic for privacy

- DNS requests or Safe Browsing queries reveal browsing history
- Telemetry data reveals client-specific information

Many applications use HTTP for performing \textit{transactional} tasks
Existing Technologies
Background

General-purpose connection-oriented proxies (CONNECT, SOCKS, Tor)

• Often includes stronger requirements and adds more overhead

• Forces a trade-off between connection setup overhead and linkability caused by long-lived connections

Application-specific protocols (private telemetry collection)

• Adds delay and requires non-trivial infrastructure
Oblivious HTTP

Protocol

Message-oriented HTTP proxy protocol for *transactional* applications

Splits knowledge of client identity and client data using a *network proxy* and *public key encryption*
1) Client discovers Target HPKE configuration
2) Client opens HTTPS connection to Proxy
3) Client sends encrypted HTTP message to Target through Proxy
4) Target decapsulates request and sends to resource server and receives response
5) Target sends encrypted response to Client through Proxy
Flow Protocol

Knows Client identity and data

Client

Proxy Server

Target Server

Resource Server

HTTPS
Flow
Protocol

The diagram illustrates the flow of data in a network setup involving a Client, a Proxy Server, a Target Server, and a Resource Server. The Proxy Server acts as an intermediary, allowing the Client to communicate with the Target Server. The Proxy Server sees the Client and Target identity, facilitating secure communication via HTTPS.
Flow Protocol

Client → Proxy Server → Target Server → Resource Server

HTTPS

Sees Proxy identity and Client data
Threat Model

Protocol

The proxy has limited trust from both client and server:

- Client trusts proxy to not leak identity to target
- Target trusts proxy to not overload it

Target compromise allows linking client messages to client identity if target and proxy collude

Traffic analysis (message size features) out of scope
Operational Considerations

Protocol

Limited to specific application protocols such as DNS

Not general purpose proxy protocol

Proxy and Target configuration discovery is out of scope

Similar to DoH

Proxies need to assist in load management

Targets need to protect against replay attacks from the proxy
Implementations

Running code

Interoperable implementations in Rust and Go

• https://github.com/martinthomson/ohttp

• https://github.com/chris-wood/ohttp-go

Test target available for interop: https://ohttp-echo.crypto-team.workers.dev
Ready for adoption?
Oblivious HTTP

draft-thomson-ohai-ohttp
Backup Slides
Open Issues

Protocol details


Additional data (https://github.com/unicorn-wg/oblivious-http/issues/70)

Anti-replay (https://github.com/unicorn-wg/oblivious-http/issues/76)
