Key Consistency and Discovery

draft-wood-key-consistency
Motivation

Background

Emerging privacy-focused protocols require a mechanism for clients to discover server public keys

- Privacy Pass: Issuer verification key
- OHTTP: Gateway public encryption key
- Tor: Relay public keys

Common requirements:

1. Unlinkability: Servers cannot link usage of a key to specific users
2. Authenticity: Clients use an authentic key for the intended server
Motivation

Unlinkability

\[ C_1 \rightarrow C_2 \rightarrow \cdots \rightarrow C_n \rightarrow S \]

\[ K \]
Motivation

Unlinkability

Sent from one of $n$ clients...
Motivation

Unlinkability
Motivation

Unlinkability

$S \leftarrow C_1 \leftarrow f(K_1) \leftarrow C_2 \leftarrow f(K_1) \leftarrow C_n \leftarrow K_1 \leftarrow K_n \leftarrow S$

Sent from $C_1$!
Motivation

Authenticity

$C_1 \rightarrow C_2 \rightarrow \cdots \rightarrow C_n$

Adv

$K' \rightarrow K \rightarrow S$
Motivation

Authenticity

$C_1\rightarrow C_2\rightarrow C_n\rightarrow Adv\rightarrow S$

$K'\rightarrow Adv\rightarrow K\rightarrow f(K)$

Adv learns something meant for $S$
Unlinkability and authenticity means that all clients in the same anonymity set have a *consistent* view of the server’s intended key, and that view is *correct*.
A key consistency and correctness system (KCCS) is something that provides consistency and correctness for clients.

KCCS varies in practice based on:

- Threat model
- Cryptographic dependencies
- Trust model and PKI
- Operational complexity
- External dependencies
Consistency and Correctness

Design space

Fetch through a trusted proxy
Fetch and verify through a trusted proxy
Fetch through multiple less-trusted proxies
Outsource to an audited or verified data store
Differing Approaches

Trusted proxy discovery

$C_i, i \in [1, \ldots, n]$ get the same $K$

Example: iCloud Private Relay key configuration
Differing Approaches

Multi-proxy discovery

Example: Consistency DoubleCheck
Differing Approaches

External database

Append-only audited log, nodes running consensus protocol, ...

Example: CONIKS for key transparency
Summary and Next Steps

Summary:

Multiple unrelated protocols and applications share the key consistency problem

All methods in the key consistency document describe architectures — not protocols — for enabling consistency

Next step:

*Adopt as informational to complement deployed solutions and proposed specs (Consistency DoubleCheck)*?
Key Consistency and Discovery
draft-wood-key-consistency
Trusted Proxy Discovery

\[ C_1, C_2, C_n \]

\[ K \]

\[ P \rightarrow S \]
Trusted Proxy Discovery

$C_i, i \in [1, \ldots, n]$ get the same $K$
Multi-Proxy Discovery

C_1
C_2
C_n
C_i

R_1
R_2
R_m

S
Multi-Proxy Discovery

\[ C_1 \quad C_2 \quad \ldots \quad C_n \quad C_i \quad R_1 \quad R_2 \quad \ldots \quad R_m \quad S \]

\[ K \quad K \quad K \quad K \quad K \quad K \]
Multi-Proxy Discovery

$C_1, C_2, \ldots, C_n$ try to lie

$S$ tries to lie

$C_i$ detects a problem!
External Database Discovery

- $C_1$
- $C_2$
- $C_n$

- $D$
- $S$

$K$
External Database Discovery

Append-only audited log, nodes running consensus protocol, …