What are Contexts

- Places with shared anonymity/privacy properties
- Privacy Pass (Current)
  - 1 Global Context
  - Client in a single anonymity set based on all the redemptions in performs
- HTTP
  - Shared Contexts
  - Site-level information boundaries
  - Top-Level Sites (First-Party)
  - Mostly
    - Cross-Site Information Transfer (3P Cookies)
- Devices
  - Shared Contexts
  - Applications
  - Mostly
    - Device Identifiers/Fingerprints
Global Redemption Context

Client:
ABCDEFGH = 10111001
Global Redemption Context

- N Issuers in the global ecosystem
- $2^N$ anonymity sets
- $(2+M)^N$ anonymity sets with M additional metadata bits
- N < 10 total issuers to maintain anonymity sets of 5000 assuming 8B people with no additional metadata (beyond the inherent present/missing bit) and one epoch for key rotations $(\log_2(8B/5000)-1)/2$. 
Sharded Redemption Context

Client: ABFG = 1000

Client: CDEH = 1111
Sharded Redemption Context

- N Issuers in the global ecosystem
- R Redemption attempts per context
  - Each redemption attempt has privacy implications
  - Where meaning of a pass may change over a period of time
- $2^R$ anonymity sets
- $(2+M)^R$ anonymity sets with M additional metadata bits
- Target number of issuers is dependent on the size of the context’s population
Redemption Context Requirements

● Strong Privacy Boundary between Contexts
  ○ Privacy leakage from redemption in one context doesn’t affect another context
  ○ Separate anonymity sets/privacy calculations
  ○ Information about redemptions in one context don’t affect a different contact.

● Unjoinable

● Application Specific Challenges
  ○ Fingerprinting (Device, IP, etc)
  ○ Dealing with leakage between contexts (cross-site tracking, caching attacks, etc)
Protocol

# context - Self-contained context for a particular set of PrivacyPass operations.
# server - Identifier for a particular known PrivacyPass issuer/server
# info - info field from the Redeem method

Client.AttemptRedemption(context, server, info) {
    if (server in redemptionContexts[context]) {
        return Redemption(server, info)
    }
    if (redemptionContexts[context].length > REDEMPTION_LIMIT) {
        return False
    }
    redemptionContexts[context].add(server)

    if (store[server.id]) {
        return Redemption(server, info)
    }
    return False
}
Issuer Stickiness

- Since the presence/absence of a token splits the anonymity set, any attempt to check if tokens are available or redeem must count against the context limits. With T total issuers, R redemption attempts, at most K redemption successes:
  - R = 1, K = 1, 1 anonymity set based on first attempted issuer in the context.
  - K = 1, R = infinity, on average T/2 anonymity sets assuming a single issuer issued tokens to this client.
  - $\sim \min(2^R, T^K)$
- Context commits to specific issuers or first R issuers requested are used.
- Stickiness Expiration
  - Never - Bad footgun.
  - Immediately - Results in attacks involving rapidly swapping through supported issuers
  - Key Rotation - privacy calculations overlapping with costs of a key rotation (most promising).
  - Data Lifetime - Linking to any other long-term data storage within the context.
  - Random Selection - Still splits the anonymity set, but less directly.
Open Questions

- Add protocol support for contexts vs leaving it purely application-layer
  ○ Latter would likely mean under PrivacyPass the anonymity set sizes would be tiny and privacy-problematic, and only solved as a result of the application-layer partitioning.

- Guidance in architecture for anonymity set/privacy math based on contexts vs global limits (#65)

- What requirements/discussion of underlying layer
  ○ unlinkability between issuance and redemption
  ○ generalized to unlinkability across contexts

- Managing issuer pinning