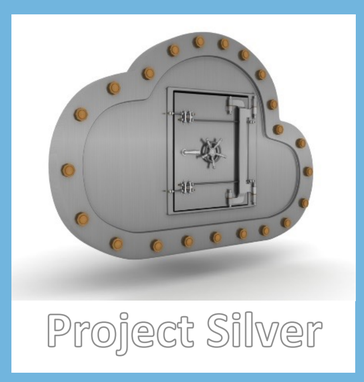


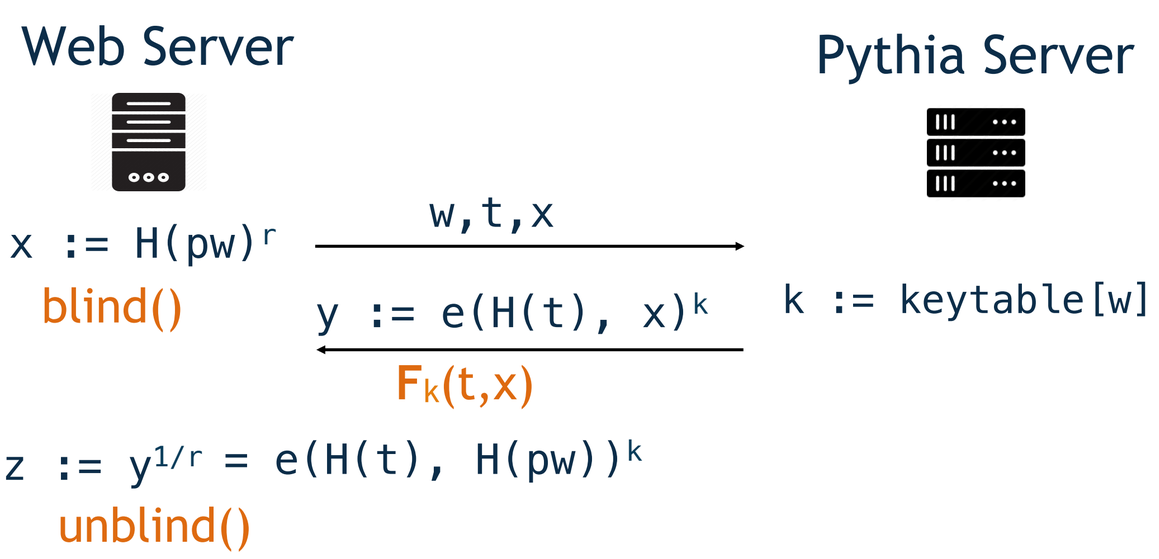
The Pythia PRF Service

A Everspaugh, R Chatterjee, S Scott, A Juels, T Ristenpart
ace@cs.wisc.edu, sam.scott.2012@live.rhul.ac.uk,
{juels, ristenpart, rc737}@cornell.edu

Goals

- Protect passwords in a variety of scenarios, including: web site authentication, password-based encryption, and Bitcoin Brainwallets
- Prevent offline dictionary attacks: force attacks online and then rate-limit guesses
- Enable key rotation so that stolen password databases can be obviated and providers can continue service
- Democratize access to this technology using a modern cloud service

Approach



- We design, and prove secure, a new primitive: Verifiable Partially-Oblivious PRF
- The Pythia PRF service sees only blinded passwords and applies a keyed PRF
- Breach of either Pythia or client is insufficient to mount offline dictionary attacks against passwords
- The scheme is key-updatable: clients or the Pythia service can rotate cryptographic keys and recover from compromise

Implementation

- We build a prototype Pythia service with modern tools: Django, Python, MongoDB, Relic crypto, and Nginx
- PRF queries as GET requests over TLS
- Open source implementation: https://github.com/ace0/pythia
- Hosted and evaluated in Amazon's EC2

Results

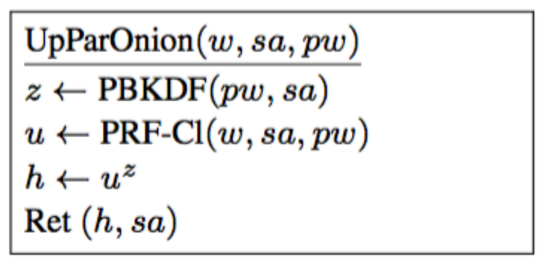
Verifying a password with Pythia:
LAN: 5.2 ms
WAN: 84 ms (WI⇆CA)

Pythia service (8-core EC2 instance)
Throughput: 1350 queries/s

Update passwords in MongoDB after key rotation:
100k: 97 s
1M: 17 mins (projected)

Password "Onion"

Combines current state-of-the art (PBKDF, bcrypt, scrypt) with Pythia



- Run PBKDF in parallel with Pythia query
- Gives defense-in-depth
- No verification latency penalty
- Client can still rotate Pythia keys