STAR: Distributed Secret Sharing for Threshold Aggregation Reporting

PPM WG, IETF 115

Shivan Kaul Sahib
Alex Davidson
Pete Snyder
Chris Wood

Idea: k-anonymity for clients reporting measurements to an untrusted server

Goals

- **Cheap**: low computational overhead and network usage for clients and servers
- **Simple**: easy to implement, well-known crypto
- **Private**: practical privacy guarantees

Central Idea: use Shamir's Secret Sharing

- Client wants to send a telemetry value to the server, but only wants the server to see it if there are >= K submissions of the same value
  - *{ "city": "Vancouver" }*
- Client generates a symmetric key by hashing its measurement
- Client encrypts its measurement using that key
- Client generates a **secret share** of that key, and sends it to the server along with encrypted message.

  —
- Iff server gets K shares of a key, it can **recover** the original key.
- Once it has the key, it can decrypt the encrypted message

Central Idea

- Use anonymizing proxy
  - OHAI
- Use Randomness Server
  - Client sends blinded input value to Randomness Server to get salt
  - To mitigate server brute-force computing all possible input values
  - Use VOPRF so Randomness Server does not learn input value

DoS attack using corrupt reports

1. Client wants to prevent recovery of a given telemetry value
2. Sends a random secret share for a given tag
3. Addressed with VSS, where the share commitments become the tag
    - VSS allows checking if a particular share is valid, even before recovery
4. Adds O(k) cost in bandwidth and computation

Implementation

- Shipping in Brave browser for telemetry
- Rust: https://github.com/brave/sta-rs
- Go: https://github.com/chris-wood/star-go/
- WASM bindings: https://github.com/brave/sta-rs/tree/main/star-wasm

What’s new in -02

- Specify verifiable and unverifiable secret sharing
- Refactor document to be easier to implement
- Add (many) more details on cryptographic APIs and functions
- Specify protocol message types for IANA
- Discuss garbage reports

Garbage reports

1. Client generates a key from message X, but encrypts and sends message Y. Recovery happens correctly, but the value will be garbage.
    a. Throwing out the batch will again cause DoS
    b. Majority vote?
2. Deterministic Blind Signatures instead of an OPRF allow the aggregation server to check which encrypted message corresponds to the right key
    a. Requires signature to be carried in encrypted message

SUPER STAR

Secret Sharing Scheme	**Signature Scheme/Protocol**	**Client threat mitigated**
Shamir Secret Sharing	OPRF	None
Verifiable Secret Sharing	OPRF	Bad shares (DoS)
Shamir Secret Sharing	Blind Signatures	Bad ciphertext
Verifiable Secret Sharing	Blind Signatures	Both

- **There seems to be strong interest in STAR**
- **We addressed feedback from the WG and it improved the document**
- **We should do this formally within the WG!**

(extra slides)

RSA Blind Signatures

1. Derive encryption key using signature: H(sign, msg)
2. Encrypt msg and signature: K(msg, sign)
3. Generate share S
4. Send S to server
5. Once server gets N Ses, it gets key
6. It decrypts to get msg and sign. It validates sign with pubk
7. It then generates the key using H(sign, msg)
8. It checks recovered key is == generated key

Central Idea: use Shamir's Secret Sharing

- Compute **symmetric key K** by hashing measurement **x**: K = H(x, rand)
- Client encrypts **x** using **K**: M = Encrypt(K, x)
- Client generates **secret share** of key: $SecretShare(K)_i$
- Client sends server: M, $SecretShare(K)_i$

- Server gets: M, $SecretShare(K)_i$
- After **N** secret shares, recover **K**: K = Recover($SecretShare(K)_{i..N}$)
- Use **K** to decrypt M: x = Decrypt(K, M)