Workgroup: Network Working Group
Internet-Draft: draft-dss-star-00
Published: 7 March 2022
Intended Status: Standards Track
Expires: 8 September 2022
Authors: A. Davidson (Brave Software)
S. K. Sahib (Brave Software)
P. Snyder (Brave Software)

STAR: Distributed Secret Sharing for Private Threshold Aggregation Reporting

Abstract

Servers often need to collect data from clients that can be privacy-sensitive if the server is able to associate the collected data with a particular user. In this document we describe STAR, an efficient and secure threshold aggregation protocol for collecting measurements from clients by an untrusted aggregation server, while maintaining K-anonymity guarantees.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 8 September 2022.

Copyright Notice

Table of Contents

1. Introduction

Collecting user data is often fraught with privacy issues because without adequate protections it is trivial for the server to learn sensitive information about the client contributing data. Even when the client's identity is separated from the data (for e.g. if the client is using the [Tor] network or [OHTTP], it's possible for the collected data to be unique enough that the user's identity is leaked. A common solution to this problem of the measurement being user-identifying/sensitive is to make sure that the measurement is only revealed to the server if there are at least K clients that have contributed the same data, thus providing K-anonymity to participating clients. Such privacy-preserving systems are referred to as threshold aggregation systems.

In this document we describe one such system, namely Distributed Secret Sharing for Private Threshold Aggregation Reporting (STAR) [STAR], that is currently deployed in production by the [Brave] browser.

2. Conventions and Definitions

The key words "**MUST**", "**MUST NOT**", "**REQUIRED**", "**SHALL**", "**SHALL NOT**", "**SHOULD**", "**SHOULD NOT**", "**RECOMMENDED**", "**NOT RECOMMENDED**", "**MAY**", and "**OPTIONAL**" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

The following terms are used:

Aggregation Server: An entity that provides some tool/software, that would like to learn aggregated data points from their user-base.

Client: The entity that uses the tool.

Measurement: The unencrypted, potentially-sensitive data point that the client is asked to report.

Message: The encrypted measurement being sent by the client.

Auxiliary Data: Arbitrary data that clients may send as part of their message, but which is not included in any security measures.

3. System Overview

3.1. Objective

In STAR, clients generate encrypted measurements, that they send to a single untrusted aggregation server. In a given amount of time, if the aggregation server receives the same encrypted value from K clients (i.e. K values), the server is able to decrypt the value. This ensures that clients only have their measurements revealed if they are part of a larger crowd. This allows the client to maintain K-anonymity, when paired with mechanisms for removing client-identifying information from their requests.

3.2. System Architecture

The overall system architecture is shown in Figure 1, where x is the measurement and aux is auxiliary data.

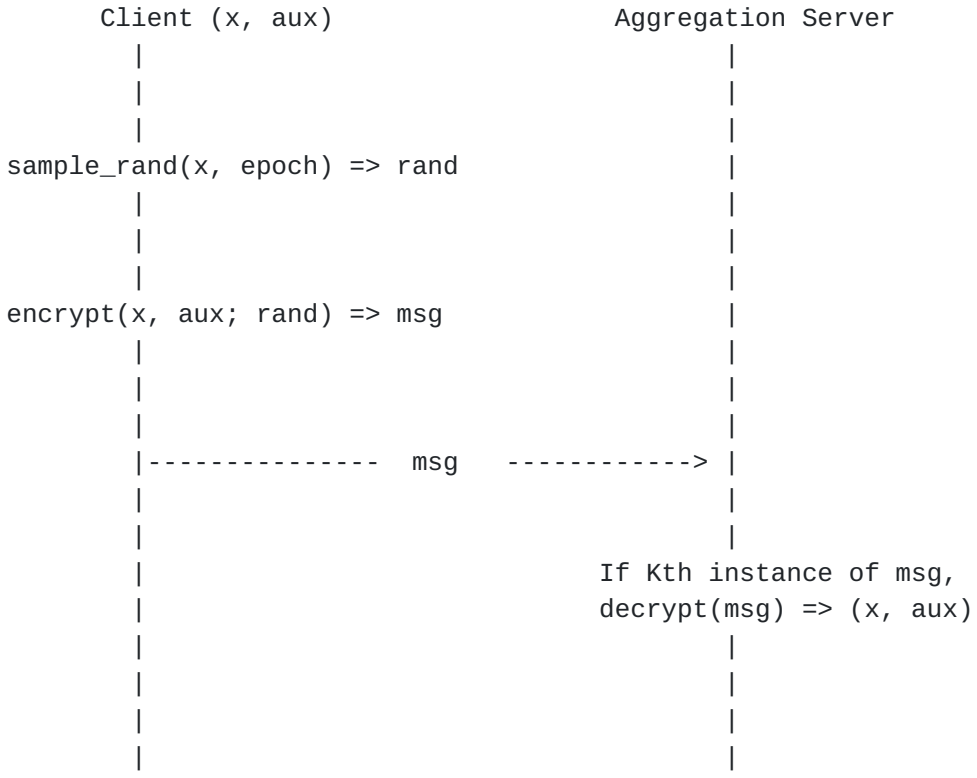

Figure 1: System Architecture

The main goal in the STAR protocol is to have the aggregation performed by a single untrusted server, without requiring communication with any other non-colluding entities. In order for the aggregation to succeed, clients must send messages that are consistent with other client messages. This requires sampling randomness that is equivalent when clients share the same measurement.

3.3. Randomness sampling

The randomness rand sampled for each message **MUST** be a deterministic function of the measurement. Either the client **MAY** sample the randomness directly by computing a randomness extractor over their measurement, or they **MAY** sample it as the output of an exchange with a separate server that implements a partially oblivious pseudorandom function protocol [OPRF]}. We discuss both cases more throughly in Section 5.1.

3.4. Measurement Encryption

The client measurement encryption process involves the following steps.

*Sample 48-bytes of randomness rand deterministically from their measurement x (as described in Section 5.1).

*The client parses rand as three 16-byte chunks: r1, r2, and r3.

*The client samples a share s of r1 from a K-out-of-N secret sharing scheme based on Lagrange interpolation, such as [ADSS]. This process involves r2 as consistent randomness for generating the coefficients for the polynomial. The client must then use independent local randomness for determining the point at which to evaluate the polynomial, and generate their share.

*The client derives a symmetric encryption key, key, from r1.

*The client encrypts the concatenation of x and aux into a ciphertext c.

*The client then generates the message to send to the server as the tuple (c,s,r3).

3.5. Server Aggregation

The server computes the output of the aggregation by performing the following steps.

*Group client messages together depending on whether they share the same value r3.

*For any subset of client messages greater that is smaller than K:

  -Abort.

*Otherwise:

  -Run secret share recovery on the set of client-received shares s to reveal r1.

  -Derive key from r1.

  -Decrypt each ciphertext c to retrieve x and aux.

  -Check that each decrypted x value is equivalent.

  -Output x and the set of all auxiliary data.

4. Comparisons with other Systems

(for information/discussion: consider removing before publication)

4.1. Private Heavy-Hitter Discovery

STAR is similar in nature to private heavy-hitter discovery protocols, such as Poplar [Poplar]. In such systems, the aggregation

server reveals the set of client measurements that are shared by at least K clients. The STAR protocol is orders of magnitude more efficient than the Poplar approach, with respect to computational, network-usage, and financial metrics. Therefore, STAR scales much better for large numbers of client submissions. Moreover, STAR allows a single untrusted server to perform the aggregation process, as opposed to Poplar which requires two non-colluding servers.

4.2. General Aggregation

In comparison to general aggregation protocols like Prio [Prio], the STAR protocol provides a more constrained set of functionality. However, STAR is significantly more efficient for the threshold aggregation functionality, requires only a single aggregation server, and is not limited to only processing numerical data types.

5. Security Considerations

5.1. Randomness Sampling

If clients sample randomness from their measurement directly, then security of the encryption process is dependent on the amount of entropy in the measurement input space. In other words, it is crucial for the privacy guarantees provided by this protocol that the aggregation server cannot simply iterate over all possible encrypted values and generate the K values needed to decrypt a given client's measurement. If this requirement does not hold, then the server can do this easily by locally evaluating the randomness derivation process on multiple measurements.

For better security guarantees, it is **RECOMMENDED** that clients sample their randomness as part of an interaction with an independent entity (AKA randomness server) running a partially oblivious pseudorandom function protocol. In such an exchange, the client submits their measurement as input, and learns rand = POPRF(sk,x;t) as the randomness, where sk is the POPRF secret key, and t is public metadata that dictates the current epoch. Sampling randomness in this way restricts the aggregation server to only being able to run the previous attack as an online interaction with the randomness server.

For further security enhancements, clients **MAY** sample their randomness in epoch t and then send it to the aggregation server in t+1 (after the randomness server has rotated their secret key). This prevents the aggregation server from being after receiving the client messages, which shortens the window of the attack. In addition, the original STAR paper [STAR] details potential constructions of POPRF protocols that allow puncturing epoch

metadata tags, which prevents the need for the randomness server to perform a full key rotation.

5.2. Cryptographic Choices

- All encryption operations **MUST** be carried out using a secure symmetric authenticated encryption scheme.

- The secret sharing scheme **MUST** be information-theoretically secure, and **SHOULD** based upon traditional K-out-of-N Shamir secret sharing.

- For functionality reasons, secret sharing operations **SHOULD** be implemented in a finite field where collisions are unlikely (e.g. of size 128-bits). This is to ensure that clients do not sample identical shares of the same value.

- Client messages **MUST** be sent over a secure, authenticated channel, such as TLS.

5.3. Oblivious Submission

Clients **SHOULD** ensure that their message submission is detached from their identity. This is to ensure that the aggregation server does not learn exactly what each client submits, in the event that their measurement is revealed. This can be achieved by having the clients submit their messages via an [OHTTP] proxy. Note that the OHTTP proxy and randomness server can be combined into a single entity, since client messages are protected by a TLS connection between the client and the aggregation server.

5.4. Leakage

Client messages immediately leak the size of the anonymity set for each received measurement, even if the measurement is not revealed. As long as client messages are sent via an [OHTTP] proxy, then the leakage derived from the anonymity sets themselves is significantly reduced.

6. IANA Considerations

This document has no IANA actions.

7. References

7.1. Normative References

[OPRF] Davidson, A., Faz-Hernandez, A., Sullivan, N., and C. A. Wood, "Oblivious Pseudorandom Functions (OPRFs) using Prime-Order Groups", Work in Progress, Internet-Draft,

draft-irtf-cfrg-voprf-09, 8 February 2022, <https://datatracker.ietf.org/doc/html/draft-irtf-cfrg-voprf-09>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/rfc/rfc2119>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/rfc/rfc8174>.

7.2. Informative References

[ADSS] Bellare, M., Dai, W., and P. Rogaway, "Reimagining Secret Sharing: Creating a Safer and More Versatile Primitive by Adding Authenticity, Correcting Errors, and Reducing Randomness Requirements", 27 June 2020, <https://eprint.iacr.org/2020/800>.

[Brave] "Brave Browser", n.d., <https://brave.com>.

[OHTTP] Thomson, M. and C. A. Wood, "Oblivious HTTP", Work in Progress, Internet-Draft, draft-thomson-http-oblivious-02, 24 August 2021, <https://datatracker.ietf.org/doc/html/draft-thomson-http-oblivious-02>.

[Poplar] Boneh, D., Boyle, E., Corrigan-Gibbs, H., Gilboa, N., and Y. Ishai, "Lightweight Techniques for Private Heavy Hitters", 4 January 2022, <https://eprint.iacr.org/2021/017>.

[Prio] Geoghegan, T., Patton, C., Rescorla, E., and C. A. Wood, "Privacy Preserving Measurement", Work in Progress, Internet-Draft, draft-gpew-priv-ppm-00, 25 October 2021, <https://datatracker.ietf.org/doc/html/draft-gpew-priv-ppm-00>.

[STAR] Davidson, A., Snyder, P., Quirk, E., Genereux, J., and B. Livshits, "STAR: Distributed Secret Sharing for Private Threshold Aggregation Reporting", 8 December 2021, <https://arxiv.org/abs/2109.10074>.

[Tor] Dingledine, R., Mathewson, N., and P. Syverson, "Tor: The Second-Generation Onion Router", 2004, <https://svn-archive.torproject.org/svn/projects/design-paper/tor-design.pdf>.

Acknowledgments

The authors would like to thank the authors of the original [STAR] paper, which forms the basis for this document.

Authors' Addresses

Alex Davidson
Brave Software

Email: alex.davidson92@gmail.com

Shivan Kaul Sahib
Brave Software

Email: shivankaulsahib@gmail.com

Peter Snyder
Brave Software

Email: pes@brave.com