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.
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 example, 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.

Randomness Server: An entity that runs an oblivious pseudorandom function ([OPRF]) service that allows clients to receive pseudorandom function evaluations on their measurement and the
The clients use the output as randomness to produce the message that is then sent to the Aggregation Server.

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.

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.
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. The client MUST sample randomness as the output of an exchange with a separate server that implements a oblivious pseudorandom function protocol [OPRF] (running in verifiable mode, i.e. a VOPRF). The original client input (i.e. the measurement) MUST be kept secret from the Randomness Server.

Note that the Randomness Server in STAR does not need to be purposely configured, providing that clients all have a consistent service that operates a VOPRF-as-a-service, in line with the functionality explained in [OPRF].

The client randomness sampling process involves the following steps:
* The client blinds the input measurement, stores state blind and sends the blinded element to the Randomness Server as rq.

* Randomness Server evaluates the blinded measurement (without learning the original measurement) and returns the evaluated element rp back to client.

* Client completes the OPRF evaluation by finalizing using original measurement x, the state blind and the evaluated element rp.

### 3.4. Auxiliary data

In Figure 1, aux refers to auxiliary or additional data that may be sent by clients, and is distinct from the measurement data protected by the K-anonymity guarantee. Auxiliary data is only revealed when the k-condition is met but, importantly, is not part of the k-condition itself. This data might be unique to some or all of the submissions, or omitted entirely. This can even be the actual measured value itself. For example: if we're measuring tabs open on a client, then the measurement being sent can be "city: Vancouver" and the aux data can be "7" for a particular client. The idea being, that we only reveal all the measurements once we know that there are at least K clients with city: Vancouver.

### 3.5. 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) in epoch t.

* 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).

* The client sends the message to the Aggregation Server via an anonymizing proxy in epoch t+1, after Randomness Server has rotated their secret key (see Section 5.1).

3.6. 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
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. STAR allows a single untrusted server to perform the aggregation process, as opposed to Poplar which requires two non-colluding servers that communicate with each other.

As a consequence, 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. See the [STAR] paper for more details on efficiency comparisons with the Poplar approach.

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.

4.3. Protocol Leakage

As we discuss in Section 5.5, STAR leaks deterministic tags derived from the client measurement that reveal which (and how many) clients share the same measurements, even if the measurements themselves are not revealed. This also enables an online dictionary attack to be launched by the Aggregation Server by sending repeated VOPRF queries to the Randomness Server as discussed in Section 5.4.1.

The leakage of Prio is defined as whatever is leaked by the function that the aggregation computes. The leakage in Poplar allows the two Aggregation Servers to learn all heavy-hitting prefixes of the eventual heavy-hitting strings that are output. Note that in Poplar it is also possible to launch dictionary attacks of a similar nature to STAR by launching a Sybil attack [Sybil] that explicitly injects multiple measurements that share the same prefix into the aggregation. This attack would result in the aggregation process learning more about client inputs that share those prefixes.
Finally, note that under collusion, the STAR security model requires the adversary to launch an offline dictionary attack against client measurements. In Prio and Poplar, security is immediately lost when collusion occurs.

4.4. Support for auxiliary data

It should be noted that clients can send auxiliary data (Section 3.4) that is revealed only when the aggregation including their measurement succeeds (i.e. K-1 other clients send the same value). Such data is supported by neither Prio, nor Poplar.

5. Security Considerations

5.1. Randomness Sampling

Deterministic randomness MUST be sampled by clients to construct their STAR message, as discussed in Section 3.5. This randomness CANNOT be derived locally, and MUST be sampled from the Randomness Server (that runs an [OPRF] service).

For best-possible security, the Randomness Server SHOULD sample and use a new OPRF key for each time epoch t, where the length of epochs is determined by the application. The previous OPRF key that was used in epoch t-1 can be safely deleted. As discussed in Section 5.5, shorter epochs provide more client security, but also reduce the window in which data collection occurs.

In this model, for further security, clients SHOULD 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 launching queries after receiving the client messages (Section 5.5). It is also RECOMMENDED that the Randomness Server runs in verifiable mode, which allows clients to verify the randomness that they are being served [OPRF].

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

The messages being submitted to an Aggregation Server in STAR MUST be detached from client 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] relay. In this flow, the Aggregation Server is configured as both the Gateway and Target Resource (the entity decrypting the message, using it, generating a response to the Encapsulated Request and encrypting the response). A separate Relay Resource is then used as to hide the client identity. Note that collusion between the Aggregation Server and the Relay Resource is expressly forbidden. All the client responsibilities mentioned in section 7.1 of [OHTTP] apply.

The OHTTP Relay Resource and Randomness Server MAY be combined into a single entity, since client messages are protected by a TLS connection between the client and the Aggregation Server. Therefore, OHTTP support can be enabled without requiring any additional non-colluding parties. In this mode, the Randomness Server SHOULD allow two endpoints: (1) to evaluate the VOPRF functionality that provides clients with randomness, and (2) to proxy client messages to the Aggregation Server. However, this increases the privacy harm in case of collusion; see Section 5.5.3.

It should also be noted that client messages CAN be sent via existing anonymizing proxies, such as [Tor], but the OHTTP solution is likely to be the most efficient way to achieve oblivious submission.

5.4. Malicious Aggregation Server
5.4.1. Dictionary Attacks

The Aggregation Server may attempt to launch a dictionary attack against the client measurement, by repeatedly launching queries against the Randomness Server for measurements of its choice. This is mitigated by the fact that the Randomness Server regularly rotates the VOPRF key that they use, which reduces the window in which this attack can be launched (Section 5.1). Note that such attacks can also be limited in scope by maintaining out-of-band protections against entities that attempt to launch large numbers of queries in short time periods.

5.4.2. Sybil Attacks

By their very nature, attacks where a malicious Aggregation Server injects clients into the system that send messages to try and reveal data from honest clients are an unavoidable consequence of building any threshold aggregation system. This system cannot provide comprehensive protection against such attacks. The time window in which such attacks can occur is restricted by rotating the VOPRF key (Section 5.1). Such attacks can also be limited in scope by using higher-layer defences such as identity-based certification [Sybil], which STAR is compatible with.

5.5. Leakage and Failure Model

5.5.1. Size of Anonymity Set

Client messages immediately leak deterministic tags that are derived from the VOPRF output that is evaluated over client measurement. This has the immediate impact that the size of the anonymity set for each received measurement (i.e. which clients share the same measurement) is revealed, even if the measurement is not revealed. As long as client messages are sent via an [OHTTP] Relay Resource, then the leakage derived from the anonymity sets themselves is significantly reduced. However, it may still be possible to use this leakage to reduce a client's privacy, and so care should be taken to not construct situations where counts of measurement subsets are likely to lead to deanonymization of clients or their data.
5.5.2. Collusion between Aggregation and Randomness Servers

Finally, note that if the Aggregation and Randomness Servers collude and jointly learn the VOPRF key, then the attack above essentially becomes an offline dictionary attack. As such, client security is not completely lost when collusion occurs, which represents a safer mode of failure when compared with Prio and Poplar.

5.5.3. Collusion between Aggregation Server and Anonymizing Proxy

As mentioned in Section 5.3, systems that depend on a relaying server to remove linkage between client messages and client identity rely on the assumption of non-collusion between the relay and the server processing the client messages. Given that STAR depends on such a system for guaranteeing that the Aggregation Server does not come to know which client submitted the STAR message (once decrypted), the same collusion risk applies.

It's worth mentioning here for completeness sake that if the OHTTP Relay Resource and Randomness Server are combined into a single entity as mentioned in Section 5.3, then this worsens the potential leakage in case of collusion: if the entities responsible for the Aggregation Server and the Randomness Server collude as described in Section 5.5.2, this results in the Aggregation Server in effect colluding with the anonymizing proxy.

6. IANA Considerations

This document has no IANA actions.

7. References

7.1. Normative References

7.2. Informative References


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