Securing ProxyStore: Encrypted Data Management for Distributed Scientific Workflows

Jordan Wels
Jwels@elon.edu
Elon University
USA

Valerie Hayot-Sasson (advisor)
University of Chicago
USA

Kyle Chard (advisor)
University of Chicago
USA

Abstract

The increased use of scientific workflows with private and protected data necessitates secure data storage and communication. ProxyStore is a system that augments distributed workflows by enabling wide-area data management via a pass by reference model. This model spans multiple endpoints that independently manage access to a variety of data. Until now, ProxyStore provides no data encryption of data stored at rest, and thus, data stored on an endpoint may be accessible via the internet. We reviewed various use cases and implemented a novel encryption approach in ProxyStore. We designed a strategy to transfer symmetric keys between endpoints leading to an easy to use encryption at rest solution for Python objects. Finally, we showed that the time needed to encrypt objects is modest and highlighted the need for further exploration of encryption policies.

ACM Reference Format:
Jordan Wels, Valerie Hayot-Sasson (advisor), and Kyle Chard (advisor). 2018. Securing ProxyStore: Encrypted Data Management for Distributed Scientific Workflows. In *Woodstock '18: ACM Symposium on Neural Gaze Detection, June 03–05, 2018, Woodstock, NY*. ACM, New York, NY, USA, 3 pages. https://doi.org/XXXXXXX.XXXXXXX

1 Introduction

Modern scientific workflows span instruments, edge devices, high performance computers, and clouds. As a result, data must be efficiently and securely moved between locations. Security is a crucially important aspect of such workflows, particularly when considering that workflows may act on protected (e.g., health care or national security data) or private research data.

Conference acronym 'XX, June 03–05, 2018, Woodstock, NY

ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/XXXXXXX.XXXXXXX

ProxyStore is a system that provides a pass-by-reference data management for distributed science workflows. ProxyStore relies on distributed data endpoints to store data and coordinate transfers between locations. These endpoints share lazy and transparent proxies of stored objects. Until now, ProxyStore has only a simple security model in which data is transferred securely, but data stored on endpoints could be accessed over the internet. Further, ProxyStore endpoints may be used to manage proxies for multiple workflows and even users at the same time. However, it does not provide isolation methods to protect data between users/workflows.

2 Motivation and requirements

We are motivated by the needs of diverse scientific workflows. For example, researchers at the advanced photon source (APS) are using beamline spectroscopy to understand the characteristics of integrated circuits. These researchers may use real-time workflows to acquire and process data which, due to commercial sensitivity, must be kept private. Similarly, federated learning workflows rely on secure exchange of model weights between distributed endpoints. Researchers have shown that model weights may release information about data distributions, which necessitates securing access to model weights.

Based on these use cases we see that it is crucial to provide encryption at rest and over the network. We also require that security be managed on a per-user or per-workflow level.

3 Proxystore

ProxyStore [1] is a system that enables just-in-time resolution of data by passing references to the object (i.e., proxies) rather than the data itself (Figure 1). ProxyStore provides various data stores and connectors to abstract storage details and communication protocols from the user. Connectors like GlobusConnector and EndpointConnector are used for wide area communication.

The EndpointConnector is ProxyStore's custom wide-area data storage and communication framework. It relies on UDP hole-punching for the peer-to-peer communication of data between sites (Figure 2). While the data is encrypted during transfer, it is publicly accessible by any other endpoint.

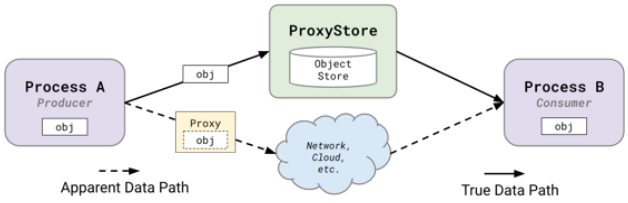

Figure 1. ProxyStore decouples the communication of object data from control flow transparently to the application. Data consumers receive lightweight proxies that act like the true object when used, while the heavy lifting of object communication is handled separately.

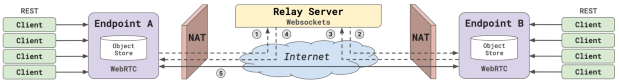

Figure 2. Client requests directed to any PS-endpoint are forwarded to the correct PS-endpoint via a peer connection. The peer connections are opened by using UDP hole-punching and a publicly accessible relay server. When PS-endpoint A wants to connect to PS-endpoint B, A asks the relay server R to forward a session description protocol (SDP) to B (1 and 2). This description contains information about how the two peers can connect, such as what protocols they support. B receives A's session description from R and replies with B's session description (3 and 4). A and B then generate interactive connectivity establishment (ICE) candidates (i.e., public IPs and ports to try for the connection) which they exchange via R. Once A and B have exchanged ICE candidates, they can connect by completing the hole punching process (5).

4 Methods

We first evaluated different encryption methodologies and selected symmetric encryption as it is considerably faster than asymmetric encryption, specifically we use AES (advanced encryption standard).

We next considered where best to integrate encryption in ProxyStore. ProxyStore exposes get/set interfaces to producers/consumers. This interface presents the logical place to add encryption. In so doing, objects will be encrypted in ProxyStore's configured storage until it is necessary to decrypt them.

Finally, we explored how to communicate symmetric keys to ProxyStore endpoints. Our goal was to limit the amount of work needed from the user, which led us to adopt asymmetric keys to securely transfer the symmetric keys. We explicitly did not want to rely on manual exchange of keys through a phone call or another physical method as it is cumbersome and error-prone. Use of asymmetric key pairs enables the automatic transfer of the symmetric keys instead of having one user put in a symmetric key to each endpoint. Figure 3 demonstrates the process of exchanging a symmetric key between two endpoints.

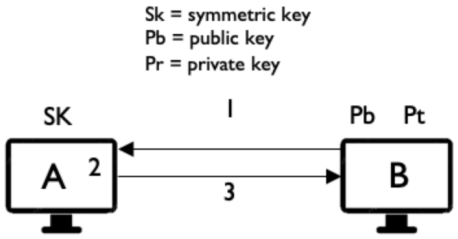

Figure 3. The step by step process of transferring the symmetric key from endpoint A to endpoint B. 1) A requests the public key and B sends it back; 2) A encrypts the symmetric key with the public key; 3) B receives the encrypted key and decrypts it with the private key.

5 Evaluation

One concern with encryption is the additional overhead incurred. Figure 4 shows the time taken to encrypt and decrypt proxies. We see only a small difference in time, likely due to the fact that these proxies are small (8 bytes). Figure 5 shows the time taken to encrypt and decrypt proxies as we increase the size of proxies. The exponential curve has a much faster increase with encryption and highlights a need to be selective when choosing to encrypt.

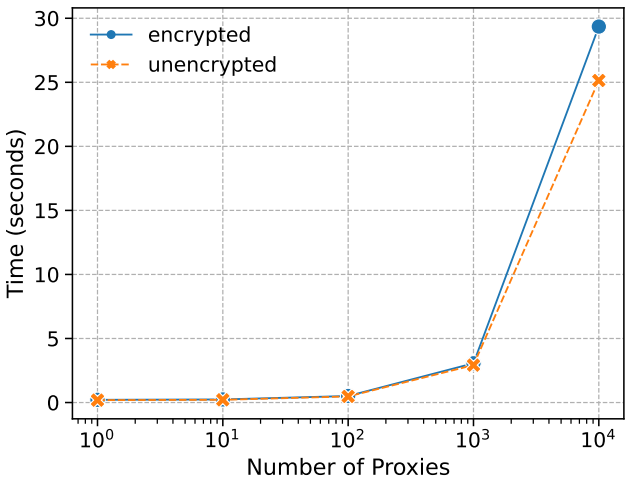

Figure 4. Time set and then get encrypted and unencrypted proxies as we increase the number of 8 byte proxies.

6 Summary and future work

We have shown that user-oriented encryption is a feasible option for ProxyStore in distributed workflows. The encryption of objects while at rest does incur overhead, however,

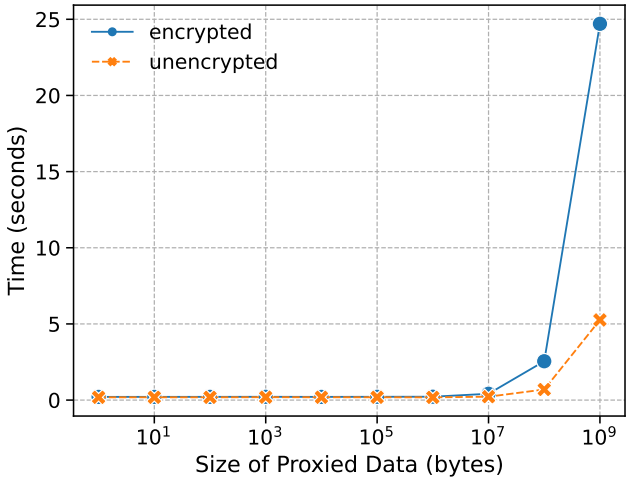

Figure 5. Time to set and then get encrypted and unencrypted proxies as we increase the size of the proxys

this is a worthwhile tradeoff if security is needed. We have also shown an easy-to-use way of automatically transferring symmetric keys. In future work we will investigate use of user-defined policies to determine what and when to encrypt.

References

[1] J. Gregory Pauloski, Valerie Hayot-Sasson, Logan Ward, Nathaniel Hudson, Charlie Sabino, Matt Baughman, Kyle Chard, and Ian Foster. 2023. Accelerating Communications in Federated Applications with Transparent Object Proxies. arXiv:2305.09593 [cs.DC]