The Secret Bunny Team

Privacy-Preserving Cloud, Email and Password Systems

Group 38

Andrew Ekstedt, Scott Merrill, Scott Russell
Sponsored by: Attila Yavuz with Thang Hoang

Motivation

Storage service offered by cloud providers bring vast benefits to human society. However, this service also brings privacy concerns to the users.

While the data privacy can be preserved by using standard encryption, it also prevents the user from querying the data on the cloud and thereby, invalidating the benefit of using cloud services.

Thus, there is a need to develop a new cryptographic primitive that allows the user to query data outsourced to the cloud, while preserving its privacy.

Algorithm Comparison

We will be comparing the different algorithms here. Discussing pros and cons of each, explaining how our implementation differed from those already implemented. It is important to also point out any abnormalities in our comparison.

This section will also discuss how our benchmarks align to the motivation behind the project. Being a focus on the research and comparison of different algorithms.

Oregon State University

Core Components

Symmetric Searchable Encryption (SSE): The SSE module provides functions to create, search, add, and delete documents to an encrypted search index.

Cloud Storage: We are using Amazon S3 as our Cloud Storage Database. This Cloud Storage database is encrypted using the scheme explained by David Cash. This is also implemented as a Storage Only Server.

Email Daemon: Using and Open Source POP3 Library this Daemon is able to update the client every minute. We did not implement a IMAP protocol simply because we do not need synchronization across devices as we are hosting the client on a single computer.

Benchmarks

We are comparing three different DSSE Algorithms against each other for our benchmarking.

1: Clusion: A Java Implementation of the Cash-DSSE Algorithm. This is primarily used as a comparison between the same algorithm, but different languages.

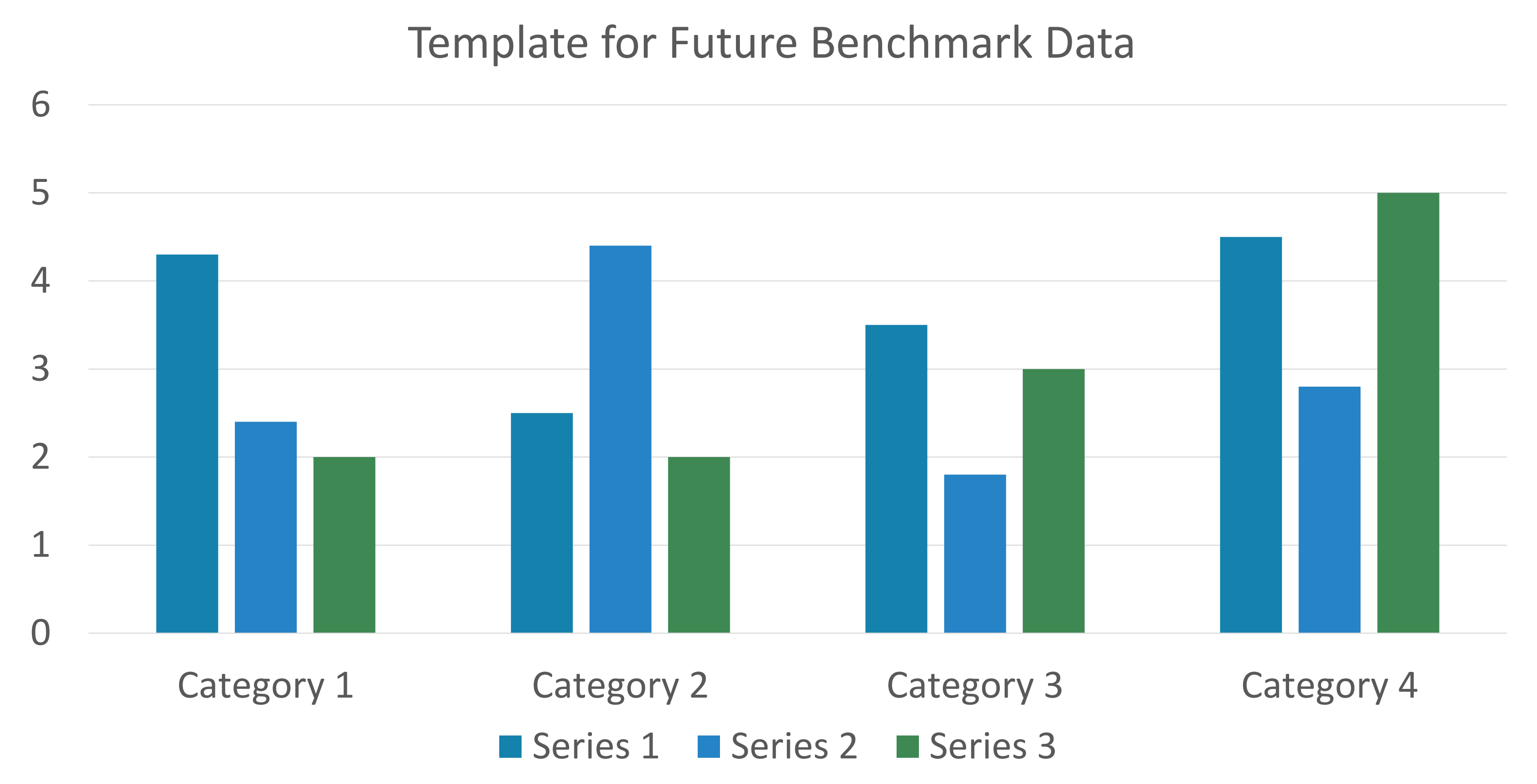
2. IM-DSSE: is a C++ implementation of a indexed matrix data structure.

3. Our Cash-DSSE Implementation: Using C++ with the same algorithm as the Clusion to benchmark against.

The Two Metrics for Benchmarking:

1. Round-Trip Delay: Testing for RTD take are preform on a slow (WIFI secure network) vs a fast (Wired Ethernet Connection) to compare how both speeds affect the bottleneck of these algorithms.

2. Index Size: The Size of the Encrypted index in Bytes.



Conclusions

The conclusion will discuss what we learned throughout the process. Including how we would change our research and implementation if we redid the project. It will focus on what did as well as things we want to do in the future if we wish to further pursue this project.

The conclusion should give a clear result from the research and benchmarking. Therefore we should determine, from our testing and research, which of the three algorithms we compared had the best overall speed, security, and usability based on benchmarking.

End.

Primary References

1. [1] D. Cash, J. Jaeger, S. Jarecki, C. Jutla, H. Krawczyk, M.-C. Rou, and M. Steiner, “Dynamic searchable encryption in very-large databases: Data structures and implementation,” Cryptology ePrint Archive, Report 2014/853, 2014. [Online]. Available: https://eprint.iacr.org/2014/853
2. [2] A. A. Yavuz and J. Guajardo, “Dynamic searchable symmetric encryption with minimal leakage and efficient updates on commodity hardware,” Selected Areas in Cryptography (SAC) 2015, Sackville, New Brunswick, Canada, August 2015, http://web.engr.oregonstate. edu/~yavuza/Yavuz DSSE SAC2015.pdf.
3. [3] “Libtomcrypt.” [Online]. Available: http://www.libtom.net/LibTomCrypt/
4. [4]T. Hoang, “IM-DSSE: Dynamic Searchable Symmetric Encryption with Incidence Matrix (IM),” 2017. [Online]. Available: https://github.com/thanghoang/IM-DSSE
5. [5] T. Moataz, S. Kamara, and S. Zhao, “Clusion: A searchable encryption library from the Encrypted Systems Lab @ Brown University.” 2017.