Usability and Trust in Cloud Encryption: Experimental Evaluation of AWS & Google SSE, CSE, and BYOK

Aahana Sharma: Stratford Preparatory Blackford High School
Faculty Mentor: Prof. Antoine Si and Mohammad Husain, Cal Poly Pomona.

Abstract

Protecting sensitive data has become a critical concern for many. Encryption still remains as a fundamental defense mechanism to secure digital information, but that alone is not enough. User-controlled encryption can limit the impact of incidents, such as breaches and could lead to loss of important data if the key is not handled carefully. In this research we experimentally evaluate the following mechanism on AWS and Google Platforms: Server Side Encryption(SSE), Client Side Encryption(CSE), and Bring Your Own Key(BYOK) to analyze the usability versus trust trade-off. The vendors we used in this experiment to test encryption were Amazon Web Services(AWS), with their S3 and Key Management System(KMS) platforms, and Cryptomator. These vendors are quite popular among companies and people who wish to keep their data secure and private, used worldwide, which is precisely why we chose them, to see just how usable the encryption is to the average person. We found that while CSE allows for more user control, it brings with it the chance of losing data due to human error. SSE is more non-expert friendly, but requires more trust in the vendor and virtually no control is with the user. BYOK allows for the user to have more control, as they create the key, but requires more expertise and coding experience, while also running the risk of losing the key and therefore the data.

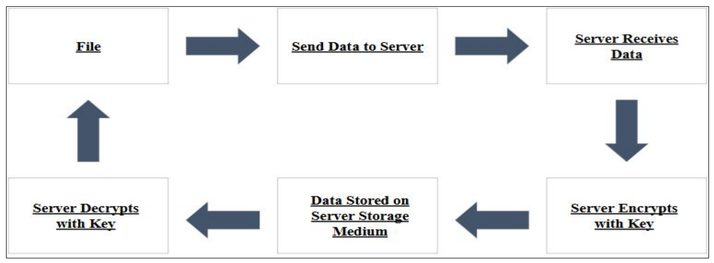

Figure 1: A diagram showing how SSE works.

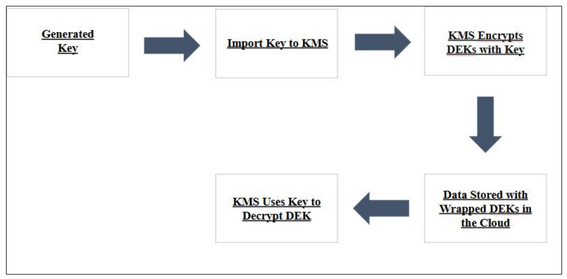

Figure 2: A diagram showing how BYOK works.

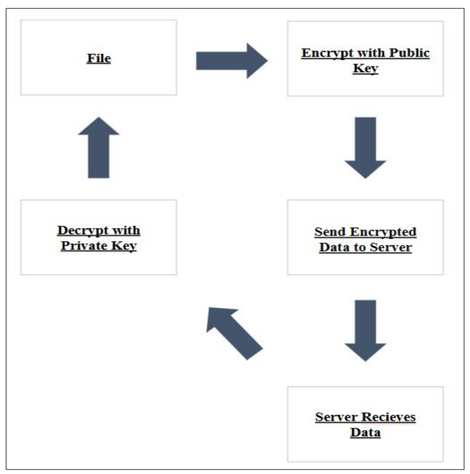

Figure 3: A diagram showing how CSE works.

Approach

To evaluate and compare the usability, effectiveness, and practical challenges of common encryption key management models in real-world cloud storage scenarios, data on how typical users handle tasks relating to keys is necessary. In order to accomplish this, basic usability data, such as encountered errors and strengths and weaknesses, as well as System Usability Scale(SUS) scores, along with even further qualitative insights on which steps were confusing and which processes were complex will be noted, along with performance logs showing how securely and reliably each encryption model protects data in a cloud context when managed by real users. The encryption models tested are the following:

- SSE-S3, where AWS handles all of the encryption
- SSE-KMS, where AWS still manages everything but allows the user more transparency
- S3+Colab, where the user separately generates the key in Google Colab and uses it to encrypt in S3
- Cryptomator, where the user uses a self-created password to use a provider generated key to encrypt their files
- AWS KMS Customer Managed Key(CMK), where the user generates a key using KMS

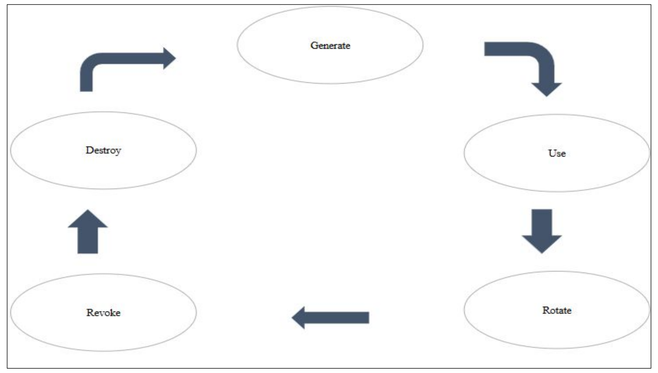

Figure 4: A diagram showing the life cycle of a key.

Tests

Service/Test	Who's generating the key	Where is the key stored	Who has access to storage
S3+Google Colab	The user	User's device	The user
Cryptomator	Cryptomator	Vault folder	The user
SSE-S3	Amazon S3	Amazon's secure S3 backend	Amazon S3
AWS KMS Amazon Managed Key	AWS KMS	AWS KMS	AWS and the user
AWS KMS CMK	The user	AWS KMS	The user and AWS

Model	Strengths	Weaknesses
SSE-S3	• Easy • Automatic • User doesn't manage keys	• Zero user control • Complete trust in AWS
SSE-KMS	• Simple • Some auditability	• Provider-controlled • Confusing for non-experts
S3 + Colab CSE	• User controls key	• Risk of losing key • Harder setup • Manual sharing
CSE(Cryptomator)	• Full client control • Zero-trust model	• Risk of losing key • Less convenient for collaboration
AWS KMS CMK	• Granular control • Key rotation • Strong audit trails	• Complex setups • Risk of locking out data • Trust in AWS

Analysis/Discussion

There were several usability and operational challenges throughout the course of this experiment. In SSE-S3, since we could not see or manage encryption keys, it was difficult to gauge true security, since it requires fully trusting AWS for key generation, storage, and rotation, which may have led to oversights of important security concepts. SSE-KMS with AWS-managed keys adds a little more transparency, but still limits control, and it was quite easy to misconfigure KMS policies accidentally, locking out access. S3 and Google Colab required us to generate, store, and share Advanced Encryption Standard(AES) keys securely, leading to a high risk of key loss or insecure distribution, along with the addition of CSE over SSE encryption making decryption steps slightly more confusing, requiring more understanding of coding languages to pull it off smoothly. Cryptomator offered full client-side control, but also demanded strong password management, and we ended up forgetting a password and losing a vault of files, which is a very common problem with CSE setups and likely one many users have run into. AWS S3 with a CMK let us have better control, but we ran into many permission issues and lost a lot of time trying to gain access, as the KMS interface was very confusing, lowering its usability.

Conclusion

Through the testing of different encryption models, from SSE to CSE, we experienced just how delicate the balance between security, control, and ease of use is. Each model offers different, unique strengths, as SSE allows for simplicity and automation while CSE lets users have complete control over their data. Distinct risks were also clear for each test, such as the complete trust that needs to be put into providers to keep user data safe, as well as operational complexity or irreversible data loss in user-controlled scenarios. To better help understand each model's usability, we used the tests to calculate the usability index from the SUS scale for each one: 82.5 for SSE-S3, 65 for SSE-KMS, 57.5 for S3+Colab, 92.5 for Cryptomator, and 52.5 for AWS KMS CMK. The results overall show that usability is just as important as cryptographic strength when encrypting in real world circumstances, since a model must be easy to use, able to recover in the event of compromised security, and trustworthy. Our next steps to further dive into the usability of encryption models would be to get user data from a wider sample, to account for many different scenarios and experiences.

References

[1] Alharbi, N., Alsiari, S., Benaissa, M., & Alaskar, H. (2023). Identification of key factors affecting data breach incidents. *PLoS ONE*, 18(5): e0284530. https://pmc.ncbi.nlm.nih.gov/articles/PMC10228450/

[2] Tucker, J. (2010). Encryption and Data Loss. *Workshop on the Economics of Information Security (WEIS)*. https://econinfosec.org/archive/weis2010/papers/session1/weis2010_tucker.pdf

[3] Fussey, P., et al. (2019). Likely Future Adoption of User-Controlled Encryption. *Carnegie Endowment for International Peace*. https://carnegieendowment.org/research/2019/04/likely-future-adoption-of-user-controlled-encryption?lang=en

[4] Khan, R., & Khan, S. U. (2023). Managing Security of Healthcare Data for a Modern Healthcare Environment. *International Journal of Environmental Research and Public Health*, 20(7): 5663. https://pmc.ncbi.nlm.nih.gov/articles/PMC10098823/

[5] McWay, D. C. (2011). Encryption and the loss of patient data. *Journal of AHIMA*, 82(1): 34-38. https://pubmed.ncbi.nlm.nih.gov/21774164/

[6] Whitten, A., & Tygar, J. D. (1999). Usability of Security: A Case Study. *People @EECS*, University of California, Berkeley. https://people.eecs.berkeley.edu/~tygar/papers/Why_Johnny_Cant_Encrypt/Usability_case_study.pdf

[7] Whitten, A., & Tygar, J. D. (1999). Why Johnny Can't Encrypt: A Usability Evaluation of PGP 5.0. *USENIX Security Symposium*. https://people.eecs.berkeley.edu/~tygar/papers/Why_Johnny_Cant_Encrypt/OReilly.pdf