Benchmarking Cryptographic Schemes for Securing Public Cloud Storages
(Practical Experience Report)

Stefan Contiu[1,2(✉)], Emmanuel Leblond[1], and Laurent Réveillère[2]

[1] Scille, 94250 Gentilly, France
{stefan.contiu,emmanuel.leblond}@scille.fr
[2] LaBRI, Université de Bordeaux, 33400 Talence, France
laurent.reveillere@u-bordeaux.fr

Abstract. Much research has focused during the last years on the security and privacy concerns of public cloud storages. Cryptographic primitives are commonly used to ensure user data confidentiality, authenticity and integrity. Confidentiality has been addressed by the use of symmetric-key encryption algorithms, while integrity and authenticity have been achieved by using message authentication codes, secure hashes or digital signatures. The choice of a specific configuration for securing an untrusted cloud storage highly depends on the expected security level, the size and type of data to store and the access pattern to these data. In this work, we are interested in overcoming the lack of comprehensive comparison of the costs and effectiveness of cryptographic primitives for securing public cloud storage, and ease an informed choice between them based on target usage conditions. We describe the results of an independent experimental study of six cryptographic schemes, representative of the principal design alternatives. Our practical experience report reveals that the best scheme for a given situation, such as a write-heavy workload of mostly small files, is not necessarily the most appropriate for a different situation such as a read-only workload of large files. We identify the scheme characteristics that are correlated with these differences and discuss the pros and cons of each design. Our experimental framework and results are available in the open for use by the community.

Keywords: Cloud storage · Security · Block cipher modes · Digital signatures

1 Introduction

Public clouds storage services such as Dropbox or Google Drive provide a convenient way for users to store and share personal data. As a result, we have witnessed a rapid adoption of these services in recent years [19]. Indeed, the cloud storage market is forecasted to grow from about $24 billion in 2016 to

L.Y. Chen and H.P. Reiser (Eds.): DAIS 2017, LNCS 10320, pp. 163–176, 2017.
DOI: 10.1007/978-3-319-59665-5_12

about $75 billion in 2021 [1]. However, despite its success, public cloud storage space is commonly assumed to be entirely untrusted, providing no guarantees over unauthorized exposure of user sensitive data. Therefore, it is not surprising that security and privacy issues in that context has gained increasing momentum within research community [24].

A traditional approach to ensure user data confidentiality, authenticity and integrity is the use cryptographic primitives. Confidentiality is addressed by the use of symmetric-key encryption algorithms, while authenticity and integrity are achieved by using message authentication codes, secure hashes or digital signatures. Cryptographic schemes are then constructed by selecting among these primitives depending on the expected level of security and privacy.

Among existing solutions, different configurations have been explored. For example, CloudProof [18] relies on AES in CTR mode for symmetric-key encryption, SHA-1 for hashing and RSA with 1024 bits key for signing. DepSky [3] uses similar cryptographic schemes except that it relies on AES in CBC mode instead of CTR. BlueSky [21] relies on AES for encryption and uses a message authentication code based on SHA-256 to provide both authenticity and integrity. In SafeSky [23] the encryption and authentication are combined by using AES in CCM mode.

Although widely used for general purpose usage, there exists very few studies comparing the costs and effectiveness of cryptographic primitives for securing public cloud storage. In this practical experience report, we are interested in overcoming this lack of a comprehensive comparison between them. We argue that the choice of a specific cryptographic construction has a direct impact on the performance and scalability of the secured cloud storage system, thus requiring a sound knowledge of its intrinsic properties. We consider different usage conditions such as various data size models and cloud workload scenarios and describe the results of an independent experimental study of six cryptographic schemes, representative of the principal design alternatives. We consider three different block cipher modes for AES encryption: chaining mode (CBC), counter mode (CTR), and an authenticated encryption mode that also covers integrity (GCM). For the public-key signature primitives, we evaluate the usage of cryptosystems based on RSA and Elliptic Curve Cryptography (ECC).

In our experiments, we perform both a set of micro-benchmarks and macro-benchmarks. Micro-benchmarks measure the intrinsic performance of a cryptographic primitive when varying the size of the cryptographic key. Macro-benchmarks assess how cryptographic primitives perform when a user interacts with a secured public cloud. We perform read and write operations on three large data sets modeled by considering different block sizes: uniform sizes, mostly small sizes, and mostly large sizes. The interaction between the user and the cloud is modeled based on four cloud workloads inspired from Yahoo! Cloud Serving Benchmark (YCSB) [7]. The workloads mimic mostly-write, write-heavy, read-heavy, and read-only operations.

The contributions of our performance comparison study aim at helping practitioners to decide which is the most appropriate cryptographic scheme for a target security level under certain usage conditions. Firstly, our results show

that there is no one-size-fits-all to security in public cloud storage. Secondly, we identify which are the schemes that better match the studied usage scenarios. Although AES in CBC in conjunction with RSA is the preferred cryptographic scheme in the literature [3,5,21], we show that other algorithms can out-perform it by a factor of 10 under specific conditions. These findings can further be used to design a cryptographic approach that changes its behavior at runtime based on contextual information.

The rest of this paper is organized as follows. Section 2 presents the cryptographic primitives we evaluate in our experiments. We describe our experimental setup in Sect. 3. Section 4 presents our evaluation results and discusses the pros and cons of each cryptographic scheme with respect to target usage conditions. Section 5 reviews related work. Finally, Sect. 6 concludes.

2 Cryptographic Building Blocks

Various cryptographic primitives are used together for ensuring confidentiality, authenticity, and integrity of user data stored in public clouds. As illustrated in Fig. 1, securing data for public cloud storage is commonly a three step process. First, the data bock is encrypted using symmetric-key algorithm (step ❶). Second, a fixed size message digest is produced by using a one-way collision resistant function (step ❷) on the input data block. Third, a digital signature algorithm is used to prove the authenticity of the message digest with respect to the user private key (step ❸). In the remainder of this section, we describe in more details each step and related cryptographic algorithms.

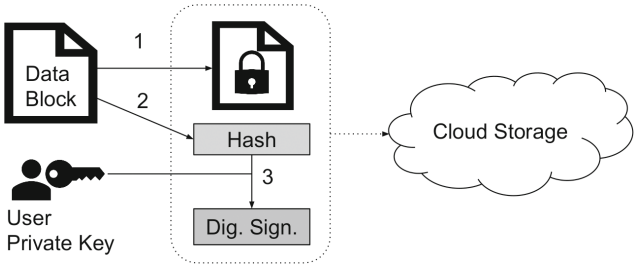


Fig. 1. Securing data for public cloud storage

2.1 Symmetric-Key Algorithms

A symmetric-key algorithm provides data confidentiality by the use of the same secret key for both encrypting and decrypting the data. Among existing algorithms, the one specified by the AES specification has become the de facto standard and is used worldwide [6]. It is a block cipher algorithm, operating on fixed-length group of 128 bits called a block with a key size of 128, 192 or 256 bits. To securely transform amounts of data larger than a block, the cipher's

single-block operation needs to be repeatedly applied accordingly to a block cipher mode. Many modes of operation have been defined [9], each one offering a different level of performance and robustness. We now describe the three major modes that we cover in our study.

CBC. Cipher Block Chaining (CBC) works by *chaining* each block to its predecessor. At each step, the current block of plaintext is `xor`-ed with the ciphertext of the previous block, and then encrypted with the secret key. Since the first block has no predecessor, a random initialization vector is used instead. The initialization vector can then be publicly stored together with the ciphertext. Due to the chaining nature of this mode, the encryption is sequential and can not be parallelized. However, because each block is `xor`-ed with the ciphertext of the previous block, not the plaintext, decryption can be parallelized. Note that the reuse of the same initialization vector can leak information only about the first block.

CTR. Counter (CTR) mode generates keystream blocks, which are then `xor`-ed with the plaintext blocks to get the ciphertext. It generates the next keystream block by encrypting successive values of a *counter*. The counter can be any function which produces a sequence which is guaranteed not to repeat for a long time, although an actual increment-by-one counter is the simplest and most popular. A nonce is combined together with the counter to produce the actual unique counter block for encryption. Since counter values at different block offsets are known, this mode can be fully parallelized. However, reusing the same nonce can leak information about all blocks, making the implementation of CTR more sensitive than CBC. Nevertheless, this mode is proven to respect tight security requirements and is formally approved by NIST [8].

GCM. Galois Counter Mode (GCM) is a block cipher mode that performs both encryption and authentication by combining counter mode and operations in a finite (Galois) field. GCM is defined for block ciphers with a block size of 128 bits. Implementing GCM can make efficient use of Carry-less Multiplication (CLMUL), an extension to the $\times 86$ instruction set used by microprocessors from Intel and AMD [11]. Similarly to CTR mode, GCM takes as input a nonce and thus reusing the same nonce with the same key leaks information about the whole message.

2.2 Message Digests

Message digests or simply hash functions are one-way collision resistant functions, mapping an input data block to a short fixed size output. The role of hash functions is to provide integrity guarantees over the data. Also, they are utilized as a preceding operation in digital signature schemes, reducing an arbitrarily large amount of data to a small output on which the signature is applied.

Hash functions work by splitting the data into fixed size blocks, and iteratively applying a compression function with an intermediate state [9]. Secure Hash Algorithms (SHA) are a class of secure hashes standardized by NIST in three family sets (SHA-1, SHA-2, and SHA-3). The first set has been proved insecure due to collision attacks [22]. The second set is a popular choice, coming with 32 and 64 bits processing variants, and producing outputs of 256, 384, and 512 bits. Lastly, the third family SHA-3 was recently standardized by NIST, not as a replacement to the previous SHA-2, but as an alternative [17].

2.3 Digital Signatures

Digital signature algorithms are employed for proving the authenticity of a data block with respect to the user private key. Moreover, they provide the properties of non-repudiation and integrity, meaning that the signing user can not deny herself as the signer and that the data block content is not altered. The verification of the message and signature pair can be openly performed by anybody knowing the user public key.

RSA is a public key cryptosystem, based on the difficult mathematical problem of factoring the product of two arbitrarily large prime numbers. The key sizes employed by RSA require a much larger length as compared to symmetric encryption, because solving the mathematical problem is faster than a brute force attack iterating over all possible keys.

Elliptic Curve Cryptography. (ECC) is a relatively novel direction in public key cryptosystems [15], that besides a considerable interest from academia, has also been integrated within technical solutions like Bitcoin, SSH, and TLS [4]. The advantage of ECC over the traditional RSA is the small nature of key sizes, implying an increase of computational speed. ECC is based on the difficult mathematical problem of discrete logarithm when the computations are performed over the points of an elliptic curve. The security of the ECC cryptosystem is highly correlated to the choice of the curve equation. Various curves have been proposed and formally reviewed, such as the ones standardized by NIST [14].

2.4 Cryptographic Strength of Key Sizes

The size of the cryptographic key is the principal factor affecting the performance and the security level of cryptographic primitives. Sufficiently large key sizes protect the cryptographic algorithms from brute force attacks on the key values. Therefore, the security strength of a cryptographic algorithm is upper-bounded by the size of the key used.

Table 1 lists three strength levels (Low, Medium, High) as specified by NIST [2]. The security strength level represents the upper bound protection

Table 1. Computational equivalence of key sizes (in bits).

Security strength	AES	RSA	ECC	SHA-2
Low	128	128	3,072	256–383	256
Medium	192	192	7,680	384–511	384
High	256	256	15,360	≥512	512

in bits for a brute force attack employed on the key values. The key sizes displayed within the same row are computationally equivalent with respect to the same security strength level. The strength for symmetric encryption is by design identical to the key size. RSA requires much larger key sizes up to 15,360 bits for a security strength of 256 bits, because solving the factorization problem is faster than a brute force attack on the key. Elliptic curve cryptography and secured hash methods require roughly the double in length.

3 Experimental Cloud-Based Data Store

In order to easily and efficiently evaluate the wide spectrum of cryptographic schemes described previously, we designed and implemented an experimental testbed, consisting of a single client accessing data on a public cloud storage. We assume that only the client can be trusted and thus data must be encrypted prior transmission to the storage node. The client component performs the actual processing and transformation (e.g., encryption, hashing) of data blocks before they are stored, as well as the reverse decoding operation (e.g., decryption, digital signature). We describe in the remainder of this section the cryptographic schemes we used in our evaluation, the model of data and the cloud workloads.

3.1 Cryptographic Schemes

We constructed six cryptographic schemes (CBC-RSA, CTR-RSA, GCM-RSA, CBC-ECC, CTR-ECC, GCM-ECC) using the main primitives described in Sect. 2. The schemes are constructed by varying the block cipher mode (CBC, CTR, and GCM) for AES symmetric encryption, and the digital signature algorithm (RSA and ECC). Message digests are generated using the SHA-2 secure hash algorithm. For each scheme, we use three different cryptographic key sizes covering the security strength levels defined in Table 1. Each key is pre-generated before the experiments using a pseudo random generator.

3.2 Data Sets

Users use cloud storage services for data files of various types among them most popular ones are photos, documents, and music [19]. Such files commonly have sizes from few hundreds of kilo bytes to several mega bytes. Smaller block sizes,

of the magnitude of tens of kilo bytes, are specific for systems that perform deduplication [16] or for modeling the entire set of files on a user machine [20]. On the other hand, larger block sizes such as 64 MB are utilized by distributed file systems operating on fixed size chunks [10]. To cover this variety of file sizes, we defined three different data sets, as depicted in Table 2, by varying the probability distribution of sizes.

Table 2. Data sets.

Data set	Probability distribution	Mean	Files	Size (GB)
Mostly-small	Log normal	256 KB	2,000	0.5
Mixed-sizes	Uniform	32 MB	100	3.1
Mostly-large	Reversed log normal	64 MB	20	1.2

The mostly-small data set follows a log normal distribution with a mean at 256 KB and contains a total of 0.5 GB of data. The mostly-large data set follows a reversed log normal distribution of files sizes for a total amount of 1.2 GB of data. The mixed-sizes data set follows a uniform distribution holding 3.1 GB of data. For all the three data sets the file sizes range from 1 KB to 64 MB.

3.3 Cloud Workloads

The ratio of read and write operations that a client performs over a cloud storage is specific to a given usage scenario. For example, when using the cloud storage to backup local files, the workload is governed by write operations. On the other hand, when sharing files such as photos with a large number of users, the workload is dominated by read operations.

To model the diversity of cloud workloads, we leverage on YCSB [7], a reference framework for benchmarking cloud storages. In addition to the three workloads defined by YCSB (write-heavy, read-heavy, read-only), we introduced a fourth one (mostly-write) composed of 5% or reads and 95% of write operations to mimic the behavior of backup scenarios. Table 3 lists the four workloads of our study and the corresponding ratios of read and write operations. The mostly-write workload performs a small number of reads (5%). The write-heavy workload consists of an even number of writes and reads. The two intensive read workloads, read-heavy and read-only, consider a small amount of writes (5%) and no writes respectively.

3.4 Implementation

Our implementation of the cryptographic schemes under evaluation relies on the open-source `openssl`[1] (v1.1.1) library. This library is implemented in a mix of

[1] https://www.openssl.org/.

Table 3. Cloud workloads

	Reads	Writes
Mostly-write	5%	95%
Write-heavy	50%	50%
Read-heavy	95%	5%
Read-only	100%	0%

C and hand-written Assembly and can take advantage of hardware acceleration provided by `AES-NI` and `CLMUL` extension instruction sets.

To test in isolation the raw performance of each cryptographic primitive, we have implemented a set of microbenchmarks in C. Our implementation uses `rtdsc` processor instruction to collect the number of cycles from the time stamp counter (`TSC`) register.

To evaluate the primitives in realistic settings, we have implemented a testbed in Python to facilitate the integration with the `cryptography.io`[2] (v1.8) the reference Python binding for `openssl`.

The cloud storage implementation contains both a Dropbox interface and a locally simulated cloud provider as an in-memory key-value store. To prevent variations of real cloud access latencies interfering with the observed outcomes and to better isolate the performance of cryptographic primitives, we report the results when utilizing the simulated cloud storage. To mimic the behavior of a public cloud storage, we added a delay of 50ms to each request to simulate a realistic round-trip latency.

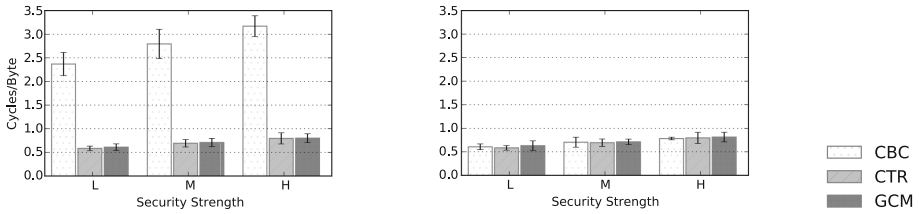
4 Results

This section presents our extensive evaluation of the previously described cryptographic schemes. We perform our experiments on a 4-Core Intel `i7-6600U` processor at 3.4 GHz with 16 GB of RAM, and operating on Ubuntu v16.04 LTS. We first test in isolation the cryptographic primitives via a set of micro-benchmarks, and we finally evaluate the primitives in realistic settings.

4.1 Micro-Benchmark

Our first set of experiments evaluate the intrinsic performance of cryptographic primitives for increasing security strength levels. In this scenario, the primitives are tested in isolation via a specialized client that sequentially perform an operation (e.g., encryption, signature) on block sizes from 512 KB to 64 MB. We repeatably execute 50 times each operation on randomly generated data and averaging the consumed CPU cycles. Our preliminary results confirm that the number of CPU cycles is always linear with respect to the size of the input data. In the remainder, we thus only show the average number of cycles per byte.

[2] https://cryptography.io/.



(a) Encryption (b) Decryption

Fig. 2. Performance of AES (cycles per byte)

Figure 2 presents our results for AES encryption and decryption. We notice that parallelizable operations have a considerable performance improvement compared to the non-parallelizable encryption in CBC mode. This large performance improvement by a factor of 4.5 for encryption is due to the pipelining technique supported by the `AES-NI` instruction set at the processor level. We also notice that the performance overhead increases almost linearly with the targeted security strength level.

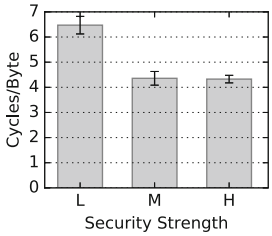
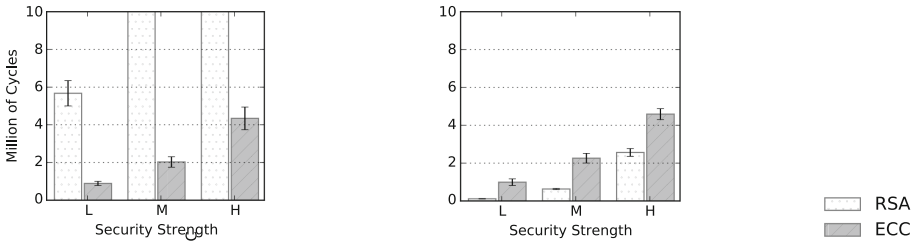


Fig. 3. Performance of SHA-2 (cycles per byte)

The cost of the SHA-2 hashing function is shown in Fig. 3. The SHA-256 method, for a low security strength level, requires on average 6.4 cycles per byte. Hashing for stronger security strength levels always perform better with an improvement of about 33%. The reason is that the calculation is done on a larger length of data at a time. Performances of SHA-384 is comparable to SHA-512, confirming that it uses the same algorithm, but truncating the hash to a smaller output. As SHA-512 offers both the higher security strength and the best performance, we use it in our macrobenchmark.

The performance results of digital signatures based on RSA and ECC are depicted in Fig. 4. Both signing and verification operations work over the secured hash produced using a hashing function such as SHA-2. Therefore, the time does not depend on the size of the input data. We thus consider only the total number of cycles required to perform the operation. The cost of the signing operation using RSA drastically increases with the size of the key. For example, the performance cost increases up to 614 millions of cycles for the strongest security



(a) Signing (b) Verification

Fig. 4. Performance of digital signature

strength level (values are truncated to 10 millions in Fig. 4a), more than 100 times the cost required for the lowest security level. On the other hand, elliptic curve signature is dramatically faster, providing a performance of 7, 58 and 153 times faster than RSA for equivalent key strengths. For the verification operation, contrary to singing, the performance of the two cryptosystems reverses. RSA performs better than Elliptic Curve, however the difference between the two is not at all as dramatic as in the case of signing.

4.2 Macrobenchmark

In this section we evaluate the cryptographic schemes in a more complex scenario that involves realistic data sets and cloud workloads, as described in Sect. 3. We measure the total time required by a client to perform all the read and write operations on the input data set. For each entry of the data set, we randomly select an operation (either read or write) to follow the probability distribution defined by the cloud workload. Figure 5 shows our results for the mostly-small test set. We can notice that RSA performs worse on mostly-write and write-heavy workloads when the security strength increases. On the other hand, read-heavy and read-only workloads do not present this trend as the verification process of the RSA signature is cheap. The CTR-ECC cryptographic scheme shows always a good performance independently of the cloud workload or the security strength.

The results of our experiments for mostly-large sizes are shown in Fig. 6. Except for RSA that performs worse with mostly-write and write-heavy workloads, we observe that differences between cryptographic schemes reduce as read operations dominate more and more the workload. We notice that RSA outperforms ECC by a insignificant factor of 2% in read-only workload. We can also notice that the performance gap between the schemes based on CBC and the ones using CTR or GCM decreases almost proportionally with the number of write operations.

Similarly to mostly-large sizes, we observe for mixed-sizes (see Fig. 7) that RSA performs worse with mostly-write and write-heavy workloads and that the gap between the different schemes tends to reduce as the number of read operation increases.

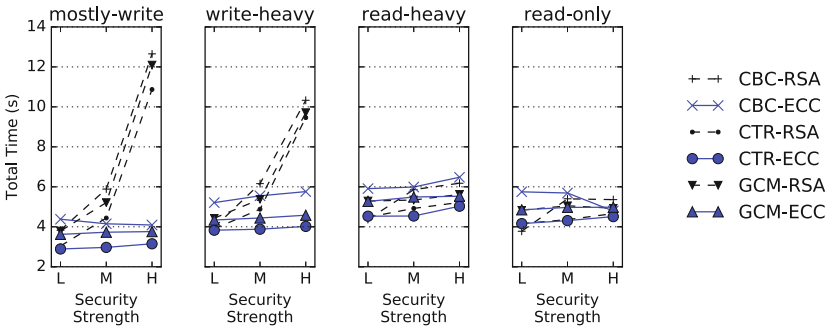


Fig. 5. Total time for small file sizes

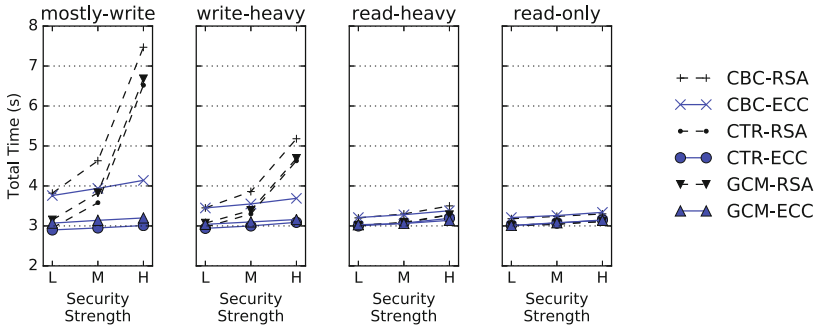


Fig. 6. Total time for large file sizes

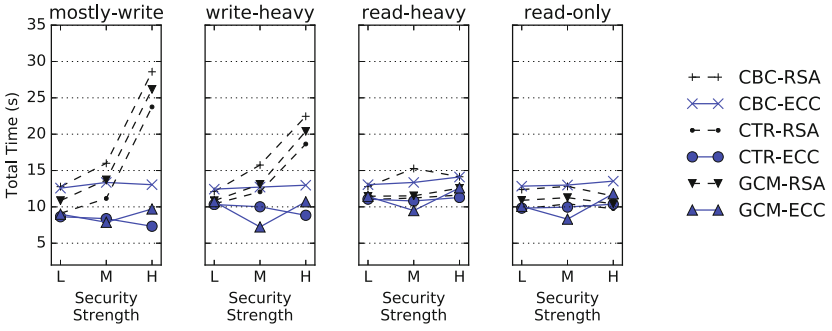


Fig. 7. Total time for uniform file sizes

4.3 Discussion

Our results show that CTR-ECC performs better in almost all usage scenarios. This scheme should be preferred if there is no prior knowledge on the workloads or data sizes.

However, we note that specific scenarios may require fine tuned schemes to maximize performances. Indeed, CTR-ECC performs better for mostly-writes and write-heavy workloads while CTR-RSA performs slightly better for read-heavy and read-only workloads. Furthermore, for read-only workloads CTR-RSA can be safely replaced by CBC-RSA, as their performances are very similar within this context. If information about data sizes is available, then a CTR-based scheme performs better as the data sizes increase.

When security is not a constraint, CTR-ECC and CTR-RSA are interchangeable as best performers. Contrary, when strong security strength is required, schemes relying on RSA should be avoided as they may induce severe performance penalties.

5 Related Work

Many previous work make use of cryptographic schemes for securing cloud storages. However, to the best of our knowledge, none of them report the result of a study to evaluate the rationale behind specific cryptographic choices. Some use of AES in CBC mode [3] while others use the CTR mode [18]. Furthermore, some even omit to describe the cipher mode they rely on [13,16]. Our benchmarking study shows that CTR outperforms CBC almost always and should be preferred. Moreover, we indicate that schemes using RSA for digital signatures [3,18] are suitable only for corner cases characterized among others by read-heavy and read-only workloads, and that ECC outperforms RSA in most usage conditions.

The costs of confidentiality, integrity and authenticity have been evaluated by Burihabwa *et al.* [5] within the cloud storage context. Besides a single cloud model, the study also considered the dispersal of confidential data over multiple storages by using erasure encoding. Although the study makes use of cryptographic primitives, there is no debate over different strength levels achieved by cryptographic keys, nor about the modeling of both the replayed test set and the read/write requests. Furthermore, the study makes use of AES in CBC mode coupled with RSA, a cryptographic scheme that, according to our findings, it is suitable only for read-only cloud workloads over mostly-small sizes.

A performance comparison study for digital signatures based on RSA and ECC has been addressed in a general context [12]. The authors propose the use of ECC for scenarios dominated by signing operations, while RSA have been proposed for scenarios dominated by verification operations. Similarly, the results of our study suggest the use of ECC for mostly-write and write-heavy workloads, and RSA for read-heavy and read-only workloads.

6 Conclusion

We have studied and compared, in this practical experience report, the performance of several cryptographic primitives that are widely used to implement security and privacy in public cloud storage. The objective of this experimental

study was to compare the costs and effectiveness of cryptographic primitives for securing public cloud storage, and not to develop original schemes.

We conducted a wide range of experiments on six different cryptographic schemes both to measure their raw speed and their performance when used in a realistic cloud storage setup. Our observations notably highlight that the best scheme for a given situation, such as a write-heavy workload of mostly small files, is not necessarily the most appropriate for a different situation such as a read-only workload of large files.

We hope that our study will bring valuable insights and guidance to other researchers interested in using cryptography techniques for data storage in the cloud.

Availability. Our experimental framework and results are available in the open for use by the community at the following webpage: https://github.com/stefan-contiu/cloud-crypto-benchmark.

Acknowledgment. This work was partially supported by Scille and DGA under contract RAPID-172906010.

References

1. Cloud storage market worth 74.94 billion USD by 2021 - MarketWatch (2016). http://www.marketwatch.com/story/cloud-storage-market-worth-7494-billion-usd-by-2021-2016-09-06-72033123
2. Barker, E.: Recommendation for key management part 1: general. Technical report, National Institute of Standards and Technology, July 2016
3. Bessani, A., Correia, M., Quaresma, B., André, F., Sousa, P.: DepSky: dependable and secure storage in a cloud-of-clouds. In: Proceedings of the Sixth Conference on Computer Systems, pp. 31–46, April 2011
4. Bos, J.W., Halderman, J.A., Heninger, N., Moore, J., Naehrig, M., Wustrow, E.: Elliptic curve cryptography in practice. In: Christin, N., Safavi-Naini, R. (eds.) FC 2014. LNCS, vol. 8437, pp. 157–175. Springer, Heidelberg (2014). doi:10.1007/978-3-662-45472-5_11
5. Burihabwa, D., Pontes, R., Felber, P., Maia, F., Mercier, H., Oliveira, R., Paulo, J., Schiavoni, V.: On the cost of safe storage for public clouds: an experimental evaluation. In: 2016 IEEE 35th Symposium on Reliable Distributed Systems (SRDS), pp. 157–166. IEEE, September 2016
6. Chown, P.: Advanced encryption standard (AES) ciphersuites for transport layer security (TLS). Technical report (2002)
7. Cooper, B.F., Silberstein, A., Tam, E., Ramakrishnan, R., Sears, R.: Benchmarking cloud serving systems with YCSB. In: Proceedings of the 1st ACM Symposium on Cloud Computing, pp. 143–154. ACM (2010)
8. Dworkin, M.: Recommendation for block cipher modes of operation: methods and techniques. Technical report, DTIC Document, December 2001
9. Ferguson, N., Schneier, B.: Practical Cryptography, vol. 23. Wiley, New York (2003)
10. Ghemawat, S., Gobioff, H., Leung, S.T.: The google file system. In: ACM SIGOPS Operating Systems Review, vol. 37, pp. 29–43. ACM, October 2003

11. Gueron, S., Kounavis, M.E.: Intel® carry-less multiplication instruction and its usage for computing the GCM mode. White Paper, May 2010
12. Jansma, N., Arrendondo, B.: Performance comparison of elliptic curve and RSA digital signatures. University of Michigan College of Engineering, April 2004
13. Kamara, S., Lauter, K.: Cryptographic cloud storage. In: Sion, R., Curtmola, R., Dietrich, S., Kiayias, A., Miret, J.M., Sako, K., Sebé, F. (eds.) FC 2010. LNCS, vol. 6054, pp. 136–149. Springer, Heidelberg (2010). doi:10.1007/978-3-642-14992-4_13
14. Kerry, C.F.: Digital signature standard (DSS). FIPS PUB 186-4, July 2013
15. Koblitz, N.: Elliptic curve cryptosystems. Math. Comput. **48**(177), 203–209 (1987)
16. Li, M., Qin, C., Lee, P.P.: CDstore: toward reliable, secure, and cost-efficient cloud storage via convergent dispersal. In: USENIX Annual Technical Conference, pp. 111–124, July 2015
17. NIST: SHA3-Standard: permutation-based hash and extendable-output functions (DRAFT FIPS PUB 202). Technical report, May 2014
18. Popa, R.A., Lorch, J.R., Molnar, D., Wang, H.J., Zhuang, L.: Enabling security in cloud storage SLAs with CloudProof. In: USENIX Annual Technical Conference, vol. 242, May 2011
19. Seybert, H., Reinecke, P.: Internet and cloud services-statistics on the use by individuals. Technical report, Eurostat, December 2014
20. Tanenbaum, A.S., Herder, J.N., Bos, H.: File size distribution on UNIX systems: then and now. ACM SIGOPS Oper. Syst. Rev. **40**(1), 100–104 (2006)
21. Vrable, M., Savage, S., Voelker, G.M.: BlueSky: cloud-backed file system for the enterprise. In: Proceedings of the 10th USENIX Conference on File and Storage Technologies, pp. 19–19. USENIX Association, February 2012
22. Wang, X., Yin, Y.L., Yu, H.: Finding collisions in the full SHA-1. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 17–36. Springer, Heidelberg (2005). doi:10.1007/11535218_2
23. Zhao, R., Yue, C., Tak, B., Tang, C.: SafeSky: a secure cloud storage middleware for end-user applications. In: 2015 IEEE 34th Symposium on Reliable Distributed Systems (SRDS), pp. 21–30. IEEE, September 2015
24. Zhou, M., Zhang, R., Xie, W., Qian, W., Zhou, A.: Security and privacy in cloud computing: a survey. In: 2010 Sixth International Conference on Semantics Knowledge and Grid (SKG), pp. 105–112. IEEE, November 2010