Secret Sharing for Cloud Data Protection

A Technical Report
presented to the faculty of the
School of Engineering and Applied Science
University of Virginia

by

Uttam Rao

May 7, 2021

Technical advisor: Yuan Tian, Department of Computer Science

Secret Sharing for Cloud Data Protection

Uttam Rao
University of Virginia
ur6yr@virginia.edu

Abstract

In 1979, Adi Shamir and George Blakley independently introduced innovative secret sharing algorithms which provide an effective method to secure private data in a distributed way. For decades, since secret sharing was introduced, it has been understood and studied by academics, but considered impractical for most large scale data security purposes due to its efficiency issues. Shamir's algorithm is used widely for a plethora of applications but, until recently, had lacked the scalability infrastructure for a large scale cloud data protection application to be feasible and cost effective. Secret sharing can be done in addition to current forms of symmetric and asymmetric encryption to provide more security guarantees than traditional data protection methods, which are still susceptible to re-encryption attacks and quantum attacks. Continued evolution of secret sharing schemes, advancements in distributed computing, and reduced costs for storage have prompted new interest in this area. Some companies, such as SplitByte Inc., have developed secret sharing based frameworks for data protection on the cloud and even have commercial pilots. This report examines the evolution of secret sharing schemes in academic research and investigates the security features of some proposed secret sharing based frameworks for cloud security applications.

1 Introduction

Cloud computing is currently one of the hottest areas. Companies of all sizes are taking advantage of the convenience, resource availability, and cost efficiency that the cloud offers. However, data security is of big concern for both companies and consumers using the cloud. For consumers, entrusting their data to companies may expose themselves to leaks of private data. Between 2017 and 2020, over 8 billion records were breached; many held sensitive data such as medical records or bank information (Puranik, 2019). For corporations, a successful cyberattack may cause millions of dollars in damage, but the impact extends far past the financial effects. Businesses that collect and store consumer data incur an average of $8.19 million in direct costs per breach, and suffer difficult-to-measure reputational damage [27, 28].

Especially at the data storage level, both traditional distributed architectures and modern cloud frameworks have many security issues. These issues are generally associated with data confidentiality, data integrity, and data availability (CIA). In particular, insider threats, re-encryption attacks such as ransomware, and quantum attacks pose security risks. For example, many of the most high profile breaches in the past year have been due to malicious ransomware attacks. In the Cognizant breach of 2020, an attacker blocked access to internal data, causing revenue to drop by 50 to 70 million dollars [29]. Because cloud environments depend on virtualization, a malicious user in a virtual environment may be able to access neighboring virtual machines or data located on the same hardware. In cloud environments, users are usually given superuser access for managing their VMs. A malicious superuser can cause a lot of havoc.

Traditional data security approaches such as data encryption, data anonymization, data redundancy, data separation, and differential privacy can generally address most data security concerns on the cloud, but none of them all at the same time [2]. The following background section details the security concerns with various cloud computing models and the current best in class methods used to protect data in the cloud. In addition to keeping their data secure, many companies want to also access it efficiently. These accesses may sometimes be through complicated analytical searches such as online analysis processing [2]. In the cloud's "pay as you go" pricing model, there is a tradeoff between data security, access efficiency, and cost.

Secret sharing, which was introduced independently by both Adi Shamir and George Blakley in 1979, is particularly useful in the context of cloud data protection. Threshold secret sharing schemes split sensitive data (the secret) into individually meaningless shares which are then distributed to n participants. In some schemes, participants can individually perform computations on shares, but the results are meaningless until a threshold number ($t \leq n$) of shares is available to reconstruct a global result. Secret sharing has been used for a multitude of applications for a long time but until recently was not of much interest for a data protection application due to space inefficiency. Over time, the continued research and development of various secret sharing schemes, such as short secret sharing and verifiable secret sharing, have made it more efficient. Traditional secret sharing assumed that the participants holding each share to be honest. Verifiable secret sharing, however, ensures that even if a participant is malicious a well defined secret can be reconstructed. Verifiable secret sharing schemes can enforce CIA all at the same time. This report examines the evolution of secret sharing schemes, investigates the use of secret sharing schemes within frameworks for cloud security applications, and evaluates their security features.

2 Background

2.1 Cloud computing models

Cloud computing is a form of effective resource sharing which offers businesses and users computer system resources on demand with high availability and scalability. Cloud service models are typically grouped into IaaS (infrastructure as a service), PaaS (platform as a service), and SaaS (software as a service). IaaS refers to services which abstract low-level physical computing resources with virtualization. IaaS cloud providers can provide these resources on demand from large stores of equipment in data centers. Providers of PaaS offer development environments to application developers. This includes a database, web server, and a code execution environment. SaaS providers offer complete application software to consumers which can be accessed through thin or thick clients.

The main cloud deployment models are private cloud, public cloud, and hybrid cloud. Private cloud refers to a cloud infrastructure operated for a single organization. It could be managed either internally or by a third party. Public cloud services are delivered on the public internet and may be shared by multiple customers. Architecturally, there is little difference between public and private cloud services, but there are many more security concerns for public cloud because resources are shared by many customers. Hybrid clouds refer to any combination of private and public clouds.

The wide variety and complexity of cloud service and deployment models provide a large attack surface and many security concerns. System breaches may result from design vulnerabilities such as improper identity credential and access management, insecure interfaces and APIs, account hijacking such as through phishing attacks, or even simple programming bugs. System breaches are especially harmful when they lead to a data breach since sensitive information may be leaked. The risk of data breaches are not unique to cloud computing, but the nature of the cloud creates a greater risk for them to occur. The private cloud is also potentially subject to any of these vulnerabilities, but the public cloud is particularly vulnerable since it is on the open web where malicious actors may be listening to traffic.

2.2 Current cloud security approaches

There are many effective techniques to protect a system and prevent a security breach. These include individual approaches to address the security issues in the previous section such as careful access management, creating secure APIs, educating employees to prevent accidental credential leakage, catching bugs in code, etc.. In the case that a system breach does allow a malicious individual to access data (data breach), there are also various methods to react and minimize the damage. However, if a system is breached, there is no way to guarantee that data is protected.

The current best in class data protection methods are various forms of encryption. Encryption uses mathematical algorithms to convert data to an unusable form that can hide it from malicious or unauthorized actors. For example, data at rest may be protected by symmetric encryption such as AES. Data in transit may be protected by hybrid encryption or public-key cryptosystems such as RSA (asymmetric cryptography). Although these classical cryptography techniques have long provided good solutions for protecting data there are still a few concerns. Firstly, both classical symmetric and asymmetric encryption are based on the idea of a "hard problem." Although in many cases these hard problems are indeed too difficult to crack, they are not provably impossible. They are not, for example, quantum proof. Secondly, the security provided by encryption is dependent on the quality of secrets management and keys being kept secret.

The current best in class methods for cloud secrets management is exemplified by HashiCorp's Vault and other similar methods. The idea is basically to centralize the secrets of a company or team in one specified server. This server applies strict identity based authorization and access controls. A user can authenticate themself with their identity, which is then mapped to the policies controlling what the user is authorized to do. The server then uses encryption when writing to a disk or communicating with users so that data is protected at rest and in motion. While this model does have very strong security guarantees, there is an inherent drawback: the central server is a single point of failure. If this server itself is successfully compromised, all security is lost. An insider with access to the unsealed server has access to all the secrets. If the server goes down, availability is lost.

Secret sharing inspired new schemes which aimed to replace this centralized system with a decentralization that can mimic something like HashiCorp's Vault for secrets management. Some implementations of systems like these can be seen in the cryptocurrency space, but the results have been mixed. Additionally, using secret sharing to split up just a key does not solve the single point of failure issue since the data could still be susceptible to ransomware or quantum attacks.

Advancements in the distributed technologies (such as the byzantine consensus) and verifiable secret sharing, however, have inspired new schemes to protect data not just by improving secrets management, but instead by splitting the data itself. These secret sharing schemes theoretically can enforce CIA all at the same time and prevent a single point of failure.

3 Related Work

Researchers have written several reports surveying the cloud data security space in peer reviewed journals or at conferences. These range from surveys about authentication and data CIA to key management, intrusion detection, and performance [1]. Researchers have also surveyed security methods on traditional distribution frameworks systems before cloud computing exploded in popularity. For example, a survey from the International Conference on Security and Management in 2012 identifies major security issues regarding data protection on distributed storage systems and classifies the various security methods at the time [3]. Some surveys also investigate data encryption and auditing on the cloud [4].

Surveys on secret sharing schemes have also been published, such as Amos Beimel's survey at the International Conference on Coding and Cryptology in 2011 where he investigates the limits of global data volume and share size [5]. However, surveys examining secret sharing for the purpose of cloud security are very few. The most recent and comprehensive survey in this area was published at the International Conference on Very Large Databases (VLDB) in 2017 by Varunya Attasena, Jerome Dermont, and Nouria Harbi [2]. Building off of work such as Beimel's, these authors aimed to wholly survey secret sharing schemes in the context of cloud data security, data access, and costs. Attasena, Dermont, and Harbi survey over 50 secret sharing schemes. They classify these schemes into eleven distinct groups to compare their security propositions and storage and computing costs. Lastly, they describe sample applications of secret sharing schemes on the cloud. Inevitably, there is some overlap between Attasena's survey and this report, such as in describing basic principles of secret sharing schemes and their properties. However, the focus of this survey is to analyze specific proposed frameworks using secret sharing schemes on the cloud (in industry and academia), including some created after the publishing of Attasena's survey in VLDB.

4 Secret Sharing Schemes

4.1 Classical schemes

There are over a dozen types of secret sharing schemes and hundreds of variations within each type. Almost all of these schemes are based on the original secret sharing schemes published by Shamir and Blakley in 1979 and the Asmuth-Bloom scheme in 1983 [6, 7, 8]. The goal of these schemes is to divide a secret S into n shares $(s_i,... s_n)$ of data such that knowing any t shares (where $t \leq n$) allows $S$to be reconstructed. Knowledge of less than t shares (the threshold) makes it theoretically impossible to reconstruct the secret. Shamir's secret sharing secures data using a random polynomial as shown in the below equations.

$$f(i) = \sum_{u=0}^{t-1} k_u \times i^u \tag{1}$$

$$s_i = f(i) \tag{2}$$

The polynomial is created over a finite set (Galois field) with coefficient k_0 as the secret and $k_u = 1,..., t - 1$as random numbers. Using this each share $s_i$is created and distributed to a participant (P_i). With the threshold number of shares, the original polynomial can be reconstructed with Lagrange interpolation over a finite set. This scheme enforces data availability even if $n - t$ participants are compromised or fail. In simpler terms, the secret is stored as a special point on a polynomial. Each participant is given a single point on this unknown polynomial. The polynomial curve can only be reconstructed when the threshold number of points are known. For example, there are an infinite number of parabolas that can pass through two points, but only one that passes through three. Since Shamir's scheme is based on polynomial interpolation, it is considered to be information theoretically secure. Unlike forms of encryption described in previous sections it is not based on a "hard problem," but on a provably impossible one.

Blakely's secret sharing achieves the same effect as Shamir's scheme, but associates each participant's share with a hyperplane (in a space with t dimensions) over a finite field instead of a point on a polynomial. The secret is the point of intersections of the hyperplanes, which can be solved using the hyperplane's system of equations. The Asmuth-Bloom scheme achieves the same effect by creating coprime integers and using the Chinese Remainder Theorem to recover the secret [8, 9].

None of the above schemes are very space efficient, especially Blakley's scheme. For a scheme to be unconditionally secure, the amount of storage required is the size of the secret times the number of shares. For small secrets this space inefficiency is tolerable, but poses huge cost issues for larger data. For example, if the size of the secret to be split up into 10 shares is 1 GB, the amount of space required is 10 GB. Another drawback of these classical schemes is that there is no verification of correctness of the shares brought together during the reconstruction process. If a participant holding shares is malicious they can cause security issues by submitting fake shares. The following two sections detail secret sharing schemes which address the drawbacks of space inefficiency and lack of verification.

4.2 Schemes for improved efficiency

Many variants of classical secret sharing schemes have been proposed for increasing efficiency, but these come at the cost of unconditional security. Some methods have tried to add symmetric encryption to secret sharing to account for this loss of security. For example, a method proposed by Krawczyk known as Secret Sharing Made Short integrates Rabin's Information Dispersal Algorithm (IDA) with Shamir's secret sharing scheme [10, 11]. Instead of the $n * s$ (where n is number or shares and s is the size of the secret) storage space that classical schemes require, IDA allows for storing the same information with $n * s/t$ space. Krawczyk's method first encrypts the data using symmetric encryption, then splits the data into n pieces using IDA with a threshold, and then also creates shares of the encryption key which are distributed to the participants (since key lengths are small the additional space this requires is minimal). The storage savings from Krawczyk's scheme still come at the cost of security. The scheme no longer has information theoretic secrecy. However, Krawczyk's scheme is *computationally* secure, which has many practical purposes. Other similar schemes and theoretical lower bounds on share size can be found in [12].

Multi secret sharing schemes also aim to improve space efficiency over classical schemes. Some are constructed similar to Shamir's, but instead of just one point on the polynomial storing secrets, many points can host secrets. [13] and [14] share data with the help of keys. In these schemes, $n$keys are used to create s shares for an amount m secrets ($m$and n must be less than s). Shares and keys are distributed to participants. To reconstruct the secret all the shares and a threshold number of keys is needed. [15] achieves a similar result without using keys. Some multi secret sharing schemes are able to achieve an overall data storage similar to the size of the secret itself. Just like short secret sharing schemes, multi secret sharing also sacrifices security guarantees for the sake of efficiency.

Some recent secret sharing schemes claim to have the efficiency of short secret sharing schemes while maintaining the information theoretic security of Shamir's secret sharing. These schemes are not yet peer reviewed. Section 5.3 of this paper investigates one such scheme created by a company called SplitByte Inc.

4.3 Verifiable secret sharing

Both classical secret sharing schemes and their variants discussed in the previous section still assume that all participants are honest and provide valid information. Verifiable secret sharing schemes solve this issue and add fault tolerance by verifying the correctness of data or keys in the reconstruction step [16, 17, 18, 19, 20, 21]. Verifiable secret sharing schemes ensure data integrity in addition to data confidentiality and data availability. There are a multitude of verifiable secret sharing schemes, all of which use some sort of homomorphic encryption combined with secret sharing. For example, [21] uses inner signatures created with a homomorphic function to verify honesty and outer signatures created by a hash function to very correctness when a secret is reconstructed. Outer signatures are used before reconstruction to verify share correctness. If at least one share is incorrect then the secret will not match with its inner signature after reconstruction. Since most verifiable secret sharing schemes support homomorphism, most also allow for data analysis on shares. [21] allows exact matches on shares, because the same key is used to share all secrets.

There are many other types of secret sharing schemes for specific purposes such as social secret sharing, proactive secret sharing, weighted secret sharing, and many more. They are detailed thoroughly in [2].

5 Secret Sharing on the Cloud

5.1 Multiple Nodes on a Single CSP

There are a plethora of different secret sharing based cloud frameworks. Some add secret sharing to classical data distribution frameworks in the cloud which distribute data over nodes (the shareholders) at a single location on a single cloud service provider (CSP). Adding secret sharing to classical distribution frameworks (surveyed in [22]) allows for improving data availability in addition to confidentiality. For example, Takahashi [23] proposes a variant of secret sharing made short designed to be suitable for the cloud. In his framework, both the dealing and reconstruction processes happen on a single master server and are distributed to nodes at one CSP. This master server could also be a node on the cloud, but it is better if the master server is at the user's side to hide all private information and keys from malicious attackers looking for shares. This type of framework still has glaring security weaknesses. For example, since all the shares are stored at the same CSP, there is still a single point of failure. If the CSP is hacked, all the shares can be collected and data can be reconstructed.

5.2 One Node Each on Multiple CSPs

Frameworks such as the one proposed in [21] address the above security issues by distributing secrets over multiple CSPs, where each CSP is a participant getting a share of the secret. It's much more unlikely that the threshold number of CSPs are all compromised at the same time, which provides better data availability. Using multiple CSPs, however, may cause data integrity issues. For example, if one CSP is compromised, it may provide an invalid share during the reconstruction process. [21] addresses this issue by using a verifiable secret sharing scheme. This also allows for data analysis on shares in a secure way without reconstruction. Frameworks, such as [21] are generally more costly since the user must deal with multiple CSPs, which may have different costs for storage. Another drawback is the access time of this type of framework is bounded by the slowest CSP.

Some frameworks combine the previous two types of frameworks to distribute secrets over multiple nodes at multiple CSPs. The schemes in these frameworks are typically some form of verifiable multi secret sharing or social secret sharing such as [24]. All three of the frameworks can be implemented on top of any existing security measures. All of the schemes and frameworks discussed so far have been from peer reviewed academic journals or conferences. With the exception of Shamir's original scheme, all of them sacrifice information theoretic security for the sake of efficiency. Some implementations in industry, however, claim to maintain efficiency while being information theoretically secure.

5.3 SplitByte's Decentralized Secret Sharing Framework

SplitByte's founder, Dr. Arvind Srinivasan, claims to have created an algorithm that provides true verifiable short secret sharing with the space efficiency of IDA and the information theoretic security of Shamir's original scheme. It uses a Repeatable Random Sequence Generator (RRSG). RRSG is a random sequence of bytes which can be reproduced using an initial state or key. In SplitByte's scheme, an RRSG byte stream is mixed with input message data which is used to provide random polynomial evaluation points for the shares to be created. Additionally, unlike Shamir's original scheme which requires a fixed field, the polynomials in SplitByte's scheme can be created over an isomorphic field. This algorithm is detailed in Srinivasan's paper [25] and SplitByte's patent [26]. It is important to note that this scheme is **still under peer review** and is not yet published in any academic journals or at conferences.

SplitByte's SplitStore framework distributes secret data over multiple nodes at multiple CSPs. Application related data and split specifications are created and stored on a distributed middleware system and all splits are stored on nodes across many CSPs. A notable difference between SplitByte and the frameworks discussed in 5.2 is that the splits are not created on a main server. Only split specifications (metadata) are created on the middleware and the actual splits happen at the data source. SplitByte also has a delegated access authentication scheme built in to its framework where users are only given credentials to the middleware. It uses a very similar authentication scheme to OAUTH PKCE (RFC 7636) and also binds an anonymous token to each user which is then recorded on each node. Although SplitByte uses a verifiable secret sharing scheme, it does not use homomorphism. It can perform any search or read only operations on shares, but it cannot update them in place (other schemes like [21] can). For new data to be inserted, all the data has to be reconstructed and then re-split. SplitByte's SplitStore product is designed for data lakes, data warehouses, or other large stores of data at rest, not for databases that may be updated frequently such as with transactions. SplitByte also claims to have a separate product, a NoSQL database system which natively embeds the SplitByte API to store transactional data, but information on this is not publicly available.

5.4 Challenges on the Cloud

There are several unique challenges posed by secret sharing schemes on the cloud that are not present with secret sharing in other contexts. Firstly, secret sharing schemes designed for the cloud must be able to deal with large amounts of data, potentially even big data volumes. Secret sharing schemes that share data all at once, such as [14], cannot handle these large volumes. Schemes such as [21], which shares individual secrets, and SplitByte's algorithm, which shares data blocks, allow for parallelizing the sharing process (potentially even in main memory), which makes sharing large data volumes efficiently possible. Secret sharing schemes on the cloud also have to be relatively space efficient. As mentioned earlier, classical schemes require $n * s$ space to securely store a secret. This may be acceptable for small pieces of data (e.g. keys) but for large amounts of data on the cloud and data streams, short secret sharing schemes or multi secret sharing schemes are necessary. In data warehouses and databases, aggregation, exact match, and update operations are commonly used. Secret sharing schemes need to not only support these operations, but also with a speed comparable to traditional systems. Some schemes, such as SplitByte's SplitStore, do not support update operations efficiently. Others, such as [21], do support in-place updates and optimize storage and query response time.

There are also challenges that are not specific to secret sharing schemes, but need to be considered in a secret sharing based cloud framework. These include issues such as bandwidth throttling by CSPs, different tiers of storage (archival, high speed retrieval, etc.), and authentication management. For example, in multiple CSP frameworks (section 5.2), dealing with different authentication methods across CSPs and accounting for different retrieval speeds from certain locations makes the frameworks complex. Because of these issues, even frameworks built on highly secure sharing schemes may be insecure due to bad architectural design and decisions. Users and companies should carefully consider the limitations of secret sharing schemes before implementing them in cloud frameworks.

6 Conclusion

When verifiable secret sharing is used as the cryptographic primitive powering data protection, there are many additional security guarantees over just encryption. With the threshold properties of secret sharing schemes, a user does not need to assume a single server is uncorrupted. Instead, this assumption is replaced with a more durable assumption that no more than the threshold number of nodes/servers are compromised at the same time. Availability guarantees also increase because fault tolerant reconstruction works even if some nodes are compromised. Shamir's secret sharing is also a cryptographic upgrade from classical heuristic cryptographic security and a single secret key securing encryption to information theoretic security with a threshold structure where no single piece of data can compromise the scheme. Of course secret sharing can be layered on top of existing forms of encryption too. They do not have to be mutually exclusive.

The main drawback of using true secret sharing schemes is space inefficiency. In order to be more space efficient, most schemes discussed in this paper compromise some level of security. The schemes are still computationally secure, but are not unconditionally secure. However, even with these more efficient schemes the cost of data storage is still significantly higher than with traditional security methods. For most companies, this cost increase may be offset by avoiding fines from violating data privacy or safe harbor laws or even reputation damage from sending notifications to users. For example, if a company stores its data in one central location and incurs a security breach on that system, there is no way to immediately prove that its data was not compromised even if the data was encrypted. Even classical distribution frameworks [22] face the same issue since each location where data is distributed still holds meaningful information. In contrast, a company can use a secret sharing based multiple CSP model (e.g. [21] or SplitByte) and split their data across, for example, five geographical locations with the threshold number at three. In this case, even if a malicious attacker is able hack a CSP and compromise a location, the data there is meaningless. If the attacker is not able to compromise three separate CSPs at the same time, then the company is able to provably prevent a data breach. Additionally, this scheme provides protection from ransomware such as re-encryption attacks since an attacker holding one or two nodes hostage is not a threat to the availability, confidentiality, or integrity of the data.

All the secret sharing schemes discussed in this paper mainly address the security of data at rest to provide an alternative to symmetric encryption, but many of them can also be applied to data in motion. For example, take the case of two users sending emails back and forth. An email can be split on the sender's device and sent over different channels to be reassembled on the receiver's device. Since the shares are sent over different channels, even if a snooper is able to compromise a channel and grab a share, they will not be able to reconstruct the email. Secret sharing schemes also provide a solution to secure data in use. Secret sharing schemes such as [21] can be the foundation for Multi Party Computation (MPC). Participants can concurrently process or compute on shares, without ever revealing the secret during computation.

For decades, since secret sharing was introduced, it has been understood and studied by academics, but considered impractical for most data protection purposes due to its efficiency issues. Continued evolution of secret sharing schemes, advancements in distributed computing, and reduced costs for storage have prompted companies to explore this space. Industry implementations of secret sharing based frameworks on the cloud are still very few, but some companies, such as SplitByte, even have commercial pilots. Given the current proliferation of data at the petabyte scale, cloud data security is now more important than ever. Secret sharing based data protection may prove to be a solution.

References

[1] Derbeko, P., Dolev, S., Gudes, E., Sharma, S.: Security and Privacy Aspects in Mapreduce on Clouds: A survey. Computer Science Review 20, 1–28 (2016).

[2] Attasena, V., Harbi, N., Darmont, J.: Secret Sharing for Cloud Data Security. The International Journal on Very Large Databases, Springer-Verlag, 2017, 26 (5), pp.657-681. Ffhal01529610f. (2017).

[3] Xu, Z., Martin, K., Kotnik, C.: A Survey of Security Services and Techniques in Distributed Storage Systems. In: 2011 International Conference on Security and Management (SAM 2011), Las Vegas, USA, pp. 3–9. (2011).

[4] Wei, D.S., Murugesan, S., Kuo, S.Y., Naik, K., Krizanc, D.: Enhancing Data Integrity and Privacy in the Cloud: An Agenda. IEEE Computer 46(11), 87–90. (2013).

[5] Beimel, A.: Secret-Sharing Schemes: A Survey. In: 3rd International Conference on Coding and Cryptology (IWCC 2011), Qingdao, China, pp. 11–46. (2011).

[6] Shamir, A.: How to Share a Secret. Communications of the ACM 22(11), 612–613. (1979).

[7] Blakley, G.R.: Safeguarding Cryptographic Keys. In: National Computer Conference (AFIPS 1979), Monval, USA, pp. 313–317. (1979).

[8] Asmuth, C., Bloom, J.: A modular approach to key safeguarding. IEEE Transactions on Information Theory 29(2), 208–210. (1983).

[9] Ding, C., Pei, D., Salomaa, A.: Chinese Remainder Theorem: Applications in Computing, Coding, Cryptography. World Scientific Publishing, Singapore. (1996).

[10] Krawczyk H., “Secret Sharing Made Short.” In: Stinson D.R. (eds) Advances in Cryptology — CRYPTO’ 93. CRYPTO 1993. Lecture Notes in Computer Science, vol 773. Springer, Berlin, Heidelberg. (1994).

[11] Rabin M. O.: Efficient Dispersal of Information for Security, Load Balancing and Fault Tolerance JACM, Vol. 36 No. 2 pp. 335-348. (1989).

[12] Beguin P., A.Cresti.: General Short Computational Secret Sharing Schemes, Advances in Cryptology - EUROCRYPT ’95, LNCS 921, pp. 194-208. (1995).

[13] Waseda, A., Soshi, M.: Consideration for Multi Threshold Multi-secret Sharing Schemes. In: 2012 International Symposium on Information Theory and its Applications (ISITA 2012), Honolulu, USA, pp. 265–269. (2012).

[14] Yang, C.C., Chang, T.Y., Hwang, M.S.: A (t,n) Multi Secret Sharing Scheme. Applied Mathematics and Computation 151(2), 483–490. (2004).

[15] Liu, Y.X., Harn, L., Yang, C.N., Zhang, Y.Q.: Efficient (n, t, n) secret sharing schemes. Journal of Systems and Software 85(6), 1325–1332. (2012).

[16] Chor B., Goldwasser S., Micali, S., Awerbuch, B.: Verifiable secret sharing and achieving simultaneity in the presence of faults, 26th Annual Symposium on Foundations of Computer Science (sfcs 1985), 1985, pp. 383-395, doi: 10.1109/SFCS.1985.64.

[17] Zhao, D., Peng, H., Wang, C., Yang, Y.: A secret sharing scheme with a short share realizing the (t,n) threshold and the adversary structure. Computers and Mathematics with Applications 64(4), 611615 (2012)

[18] Feldman, P.:A practical scheme for non-interactive verifiable secret sharing, 28th Annual Symposium on Foundations of Computer Science (sfcs 1987), 1987, pp. 427-438, doi: 10.1109/SFCS.1987.4.

[19] Shi, R., Zhong, H., Huang, L.: A (t, n)-threshold verified multi-secret sharing scheme based on ECDLP. In: International Conference on InterSoftware Engineering, Artificial Intelligence, Networking, and Parallel/Distributed (ACIS 2007), pp. 9–13 (2007)

[20] Attasena, V., Harbi, N., Darmont, J.: fVSS: A New Secure and Cost-Efficient Scheme for Cloud Data Warehouses. In: ACM 17th International Conference on Data Warehouses and OLAP (DOLAP 2014), Shanghai, China, pp. 81–90. ACM (2014)

[21] Attasena, V., Harbi, N., Darmont, J.: A Novel MultiSecret Sharing Approach for Secure Data Warehousing and On-Line Analysis Processing in the Cloud. International Journal of Data Warehousing and Mining 11(2), 21–42 (2015)

[22] Padmanabhan, P., Gruenwald, L., Vallur, A., Atiquzzaman, M.: A survey of data replication

techniques for mobile ad hoc network databases. The VLDB Journal 17(5), 1143–1164 (2008)
[23] Takahashi, S., Iwamura, K.: Secret Sharing Scheme Suitable for Cloud Computing. In: International conference on Advanced Information Networking and Applications (AINA 2013), Barcelona, Spain, pp. 530–537 (2013)
[24] Zheng, T., Wu, H., Lin, H.W., Pan, J.: Application of belief learning model based socio-rational secret sharing scheme on cloud storage. In: 6th International Conference on Genetic and Evolutionary Computing (ICGEC 2012), Kitakyushu, Japan, pp. 15–18 (2012)
[25] Chan, C., Srinivasan, A.: Short Secret Sharing Using Repeatable Random Sequence Generators. (2018). Retrieved from https://arxiv.org/ftp/arxiv/papers/2101/2101.09317.pdf
[26] SplitByte Inc., 2021. Systems and Methods for Managing Data Based on Secret Sharing. (Apr. 2021). US Patent No. 10985911 B2. Filed Nov. 13th, 2018, Issued Apr. 20th, 2021.
[27] Puranik, M.: Council Post: What Is the Cost of a Data Breach? In: Forbes. Retrieved from https://www.forbes.com/sites/forbestechcouncil/2019/12/02/what-is-the-cost-of-a-data-breach/?sh=39250a8a29e7. (2019).
[28] Columbus, L.: 2020 Roundup of Cybersecurity Forecasts & Market Estimates. In: Forbes. Retrieved from https://www.forbes.com/sites/louiscolumbus/2020/04/05/2020-roundup-of-cybersecurity-forecasts-and-market-estimates/?sh=279f4448381d. (2020).
[29] Cimpanu, C.: Cognizant expects to lose between $50m and $70m following ransomware attack. Retrieved from https://www.zdnet.com/article/cognizant-expects-to-lose-between-50m-and-70m-following-ransomware-attack/. (2020).