CoProtect: Collaborative Management of Cryptographic Keys for Data Security in Cloud Systems

Lorenzo Bracciale[1], Pierpaolo Loreti[1], Emanuele Raso[1], Maurizio Naldi[2] and Giuseppe Bianchi[1]
[1]*Department of Electronic Engineering, University of Rome Tor Vergata, Rome, Italy*
[2]*Department of Computer Science and Civil Engineering, University of Rome Tor Vergata, Rome, Italy*

Keywords: Cloud Computing, Security, Pedersen Key Distribution, Shamir Secret Sharing.

Abstract: Cryptography key management system plays a very central role in the cloud data security. Nonetheless, a great part of the current commercial solutions rely on cloud providers that hold both the encrypted data and the related private master key of their served customers in their secure key vaults, having a de-facto total control on their customer digital assets. Conversely, entrusting customer companies for key holding can be dangerous as witnessed by many cases of key loss or theft. In this work we present CoProtect, a novel architecture to protect the cryptography keys in cloud systems that leverage on the cooperation between the cloud provider and the customer company. With such trust model, we present the proposed data management strategy, the key generation and the crypto procedures, and a proof of concept.

1 INTRODUCTION

Cloud will drive 83% of enterprise workloads by 2020, but, according to a Microsoft survey, over 90% of the public and business leaders recognize data as the most critical company asset and are worried about its security, availability and privacy in the cloud (Subashini and Kavitha, 2011), (Singh and Chatterjee, 2017). Among all the concerns, security is the top inhibitor to cloud adoption and, according to 451 Alliance survey data, current cloud security systems strive to provide the requested solutions to fill the real and perceived security market needs (Alliance, 2019). One of the reasons of this gap is accountable to the nature of "cloud services", a generic term that conceals a wide set of cross-domain and technologically heterogeneous services.

In this context, it is difficult to design and adopt a single security architecture, as one size does not fit all, with different technological domains demanding for different, dedicated and customized security solutions (Song et al., 2012). Moreover any proposed security measure must not impair the rapid development and maintenance which are among the main reasons why companies choose cloud offers. This market is also fostered by the new General Data Protection Regulation (GDPR), the legal framework that sets guidelines for data protection in Europe. While big cloud providers such Amazon, Microsoft or Google develop proprietary security systems integrated in their infrastructure, recently third parties also propose different ways offering Data Protection As a Service (Vu et al., 2015).

Policy management, authentication through certification and encryption are among the most effective techniques adopted in cloud security systems, but several problems are still far from being solved. For instance, privacy issues can occur in case of Electronic Health Records storage, where the use of certificates unnecessarily reveals the identity of their holders (Sajid and Abbas, 2016) (Yüksel et al., 2017) (Ben-Assuli, 2015). A common technique to enforce data protection is to use comprehensive encryption at file level (Sood, 2012), so to move from the protection of the document contents to the management and the protection of the cryptographic keys (Rittinghouse and Ransome, 2017). While this allows more flexible mobility services and data replication, new problems arise when dealing with the key management. In particular, in this work we focus on a very basic and simple problem: *who holds the master key?* That is a fundamental question whose answer may not be so obvious. Is it better to totally entrust the cloud providers by giving them both the customer (encrypted) data *and* the master key, or keep the master key inside the customer company?

This latter solution is currently offered by many cloud providers and referred to as *Hold Your Own*

Bracciale, L., Loreti, P., Raso, E., Naldi, M. and Bianchi, G.
CoProtect: Collaborative Management of Cryptographic Keys for Data Security in Cloud Systems.
DOI: 10.5220/0008921603610368
In *Proceedings of the 6th International Conference on Information Systems Security and Privacy (ICISSP 2020)*, pages 361-368
ISBN: 978-989-758-399-5; ISSN: 2184-4356

Key (HYOK) scheme (Nickel, 2019). However, with HYOK, especially when small and micro enterprises are taken into account, the system provides a weaker protection of the key with respect to case where big cloud providers with security professional teams and dedicated hardware module (e.g., based on Trusted Platform Modules) are demanded to store the keys on behalf of the customer companies. Indeed, if key theft or loss events occur, cloud providers do not have any means to recover data content. An example of this was the recent case when, after the CEO death, over 190 millions of dollars were trapped in a wallet of an exchange because nobody except him knew the password (Rushe, 2019). Moreover, cloud-based services and applications cannot reason over the encrypted data for giving searching and analytics services.

For this reason cloud providers, more or less explicitly, push companies to use key management services offered by them, either with *Bring Your Own Key* (BYOK) scheme, or just by generating and holding their master key on behalf of their customers. This solution has the drawback of exposing the data to additional risks, comprising the potential risk of vendor lock-in, and subtracting data owners from the full control and access on their data. Moreover, in some cases, holding encrypted data together with the related key can be assimilated to having the data in clear and not using encryption at all.

The envisaged goal of this work is to provide a third option, based on the *collaboration* between companies and cloud providers, that on one hand gives the companies the control of their data, and on the other hand offers disaster recovery and protection against accidental loss or key theft by any of the actors. The presented architecture, called CoProtect, allows companies to be the sole responsible for their data disclosure and foster the construction of data access and modification logging service (required, in some cases, to by GDPR), other than implementing their own access control policies independently. The rest of the paper is organized as follows. Section 2 reviews the related work, section 3 presents the related background techniques (Pedersen distributed key generation, ElGamal crypto-system). Section 4 presents the proposed solution with the related trust model and the description of architecture and procedures needed to implement the solution. Section 5 presents a proof-of-concept implementation and finally conclusions are drawn.

2 RELATED WORK

2.1 Crypto-based Solution

Data security in the cloud is usually based on encryption systems that however may inhibit many of the advantages offered by the cloud. Many works proposed solutions to provide typical cloud services on encrypted data, such as access control, document search and multi-user collaboration. For example, a system based on searchable encryption has been proposed in (Singh and Batten, 2017) and enables private querying on public data so that the contents of user queries and their replies are private. In (Huang et al., 2017), authors propose a cryptographic system to support shared read and write access to files in the cloud. The system is based on Ciphertext-Policy Attribute-Based Encryption (CP-ABE) (Bethencourt et al., 2007) and allows a fine grain control on users. Differently, this work is focused on implementing a trust model for the cryptographic key management rather than on providing services on top of encrypted data; however these kind of techniques can be used to foster "blinded" services on top of the encrypted data, but this application is outside the scope of this work. To enable data sharing, in (Zuo et al., 2017) Zuo *et al.* propose a technique based on a central authority that issues keys according to the user's attribute set. Similarly to our work they divide the key into two parts to enhance the system security: one part is stored in a potentially insecure place, the other in a physically safe place. To protect data association, De Capitani di Vimercati *et al.* (De Capitani di Vimercati et al., 2013) propose a system of fragmentation and encryption to improve privacy. Other solutions to securing data storage on a database, has been proposed by (Ganapathy et al., 2019), where the authors designed a management system and a key data storage based on the Chinese Remainder Theorem for the management of user groups access policies.

2.2 Commercial Cloud Platform Privacy Solution

To ensure the security and privacy of user data, commercial cloud service providers (CSP) must comply with regulations imposed by national and international bodies. In Europe, for example, Cloud Infrastructure Services Providers in Europe (CISPE) defines some minimum security requirements (CISPE, 2019), recommending specific solutions such as:

- data encryption capabilities available in storage and database services;

- flexible key management allowing the customers to choose whether to have the CSP manage the encryption keys or enable the customer to keep complete control over keys;
- provide APIs for customers to integrate encryption and data protection in the offered services.

Furthermore, the European Community has defined the General Data Protection Regulation (GDPR), a specific privacy framework which commits CSPs to a systematic privacy assessment. Some works proposed GDPR compliant cloud solutions such as the MUSA project (Rios et al., 2019) that proposes a privacy by design novel cloud architecture based on service level agreements. However, complete solutions are difficult to apply in commercial clouds that are often based on outsourcing and service composition. Commercial clouds are starting to provide solutions to manage data privacy. For example, Microsoft Azure Information Protection provides a solution to classify and optionally protect documents and emails by applying labels. The system is based on the encryption of a content with a symmetric key which is encrypted with an asymmetric key and prefixed, together with the access policies, as a header of the encrypted content. Microsoft, holding the private key, can decrypt the header and, once the access policies have been verified, return the symmetric key to the user needed to decrypt the content (Microsoft, 2019).

2.3 Domain Specific Solutions

Privacy solutions can present domain specific challenges. For instance, in the Big Data field, data is typically distributed and the main issue is related to assuring security and privacy during the processing. In particular, the distributed nature of the cloud and the continuous movement of data implies that security cannot just rely on infrastructure (data centers) and end-to-end data tunnels (e.g. TLS), but it must be included in the data itself, fostering a data-centric control model. In (Puthal et al., 2017) a selective encryption method has been presented to ensure and maintain the confidentiality of Big Data flows according to the related data sensitivity. The solution is based on the presence of a common key shared by all the peers, which is initialized and updated by an agent called Data Stream Manager. In (Pasquier et al., 2016), the authors propose Information Flow Control, a data-centric mechanism to implement access control by complex policies. The technique is based on the trust between the data owner and the cloud provider and thus cannot guarantee the full data privacy. Personal health data represents another special case for data management in the cloud since data privacy must be guaranteed, but it should be available to support medical research works (Singh and Batten, 2017) and it should be readily available by health professionals in case of emergency (Yüksel et al., 2017). Moreover privacy involves not only the content of the health data but also the presence of data related to some people inside healthcare databases (Coats and Acharya, 2014).

3 BACKGROUND

In this section we provide the necessary technical background that will be used in the proposed solution.

3.1 Pedersen Distributed Key Generation

In his seminal work (Pedersen, 1991) Torben Pedersen presents a protocol in which n parties jointly create a public key for the ElGamal cryptosystem, and obtain a "part of" the private key so that every group of t-over-n parties can collaborate to decrypt any message encrypted with that public key without any party knowing the full private key. We briefly recap the underlying methodology as we used this technique for key generation. The i-th entity constructs a polynomial of degree $t-1$ with random coefficient $a_{k,i}$ as:

$$p_i(x) = a_{0,i} + a_{1,i}x + a_{2,i}x^2 + \ldots + a_{t-1,i}x^{t-1} \mod q$$

where t is the number of parties needed to decipher a message, while q is a large prime. Then, the i-th entity distributes a point in that polynomial to the other $n-1$ parties; specifically it sends $p_i(x_j)$ to the j-th entity, with $i \neq j$.

Once it receives the points from all the other entities, i-th entity can calculate its "full share" as:

$$y_i = \sum_{j=1}^{n} p_j(x_i)$$

In this way parties are collectively creating this polynomial:

$$P(x) = \sum_{i=1}^{n} a_{0,i} + \sum_{i=1}^{n} a_{1,i}x + \ldots + \sum_{i=1}^{n} a_{t-1,i}x^{t-1} \mod q$$

The private key, unknown to every party, is:

$$s = \sum_{i=1}^{n} a_{0,i}$$

but all the party can calculate their share of the secret key starting from their full share as:

$$s_i = y_i \Lambda_i \mod q$$

where $\Lambda_i = \prod_{shares\ x_k \neq x_i} \frac{-x_k}{x_i - x_k}$ is the Lagrange multiplier, and $s = \sum_{shares\ x_i} s_i \mod q$.

The public key can be collectively created as:

$$g^s = \prod_{i=1}^{n} g^{a_{0,i}} = g^{\sum_{i=1}^{n} a_{0,i}} \quad (1)$$

where g is the primitive root of a strong prime p, i.e. $p = 2q+1$ and $g = (l)^2 \mod p$, where l is a random integer.

3.1.1 ElGamal Encryption System

ElGamal is an asymmetric key encryption algorithm for public-key cryptography, whose complexity is based on the calculation of the discrete logarithm. Given its homomorphic proprieties, it is well suited to be used with Pedersen Distributed Key Generation. Indeed, as explain below, by construction ElGamal allows the chaining of multiple partial decryption operations performed by all the parties, simply by multiplying their decryption results. This allows not only to ultimately obtain the fully decrypted text without that any party acquire information on that, but also preserve the parties from sharing information on their private key fragments.

Encryption. Given the generator g, the private key s and a prime p, as previously described, the encryption process chooses a random number $r \in [1, p-1]$. Then, it encrypt a message m as the pair $(c_1,\ c_2)$, where

$$c_1 = g^r \mod p$$
$$c_2 = mg^{sr} \mod p$$

Decryption. Given the pair (c_1, c_2), decryption happens calculating:

$$k = c_1^s \mod p = g^{rs} \mod p$$

and then:

$$c_2 k^{-1} \mod p = mkk^{-1} \mod p =$$
$$= mg^{sr} g^{-rs} \mod p = m \mod p$$

where k^{-1} is the multiplicative inverse of k modulo p.

Decryption with Pedersen Keys. In spite of s, each party uses its secret share s_i, applies the decryption algorithm and passes to the result to another party. After t parties, we have the full decryption of the original message, i.e.:

$$c_2 k_1^{-1} \cdot \ldots \cdot k_t^{-1} \mod p = m \cdot g^{sr} \cdot \prod_{i=1}^{t} g^{-rs_i} \mod p =$$
$$= m \cdot g^{sr} \cdot g^{-r\sum_{i=1}^{t} s_i} \mod p = m \cdot g^{sr} \cdot g^{-rs} \mod p =$$
$$= m \mod p$$

The message has been deciphered without s being revealed to any of the parties.

4 CoProtect

In this section we present the proposed trust model and why it is different from other industrial options. We also present the architecture of the proposed solution together with the the technical operations and data structure needed to implement it on the cloud.

4.1 Trust Model

The trust model, with reference to the entity holding the master key, is the key feature where the proposed system mainly differs from the great part of commercial and literature solution. As said, we want to avoid totally relying on either the cloud provider or the company/customer: in the former case we would have the control on the data totally assigned to the cloud provider, while in the latter case we would not have any mechanism to prevent from accidental key loss or key theft, by which especially small companies can be affected. A third option would be to use a Trusted Third Party (TTP) appointed, for example, to act in the case of disaster recovery. Entirely relying on a Trusted Third Party is a strong assumption and represents an important security bottleneck that should be avoided, whenever possible, especially if we are dealing with multiple companies and very sensitive data. For this reason, we started from alternative solutions that have been presented in the literature to relax the trust assumption and replace the need of a single Trusted Third Party with a "Simulated TTP", e.g., in Sepia (Burkhart et al., 2010) and P4P (Xie et al., 2008). In our case the Simulated TTP is given by the collaboration between company and cloud provider, as described hereafter.

4.1.1 Multisignature Scheme

With reference to figure 1 we have a multisignature scheme based on three different keys: the first held by the cloud provider, the second by the company, while the third is encrypted and *protected with the customer password* and stored on both the parties. These keys are associated to a public key so that everybody can encrypt any data with the public key, but two out of the three keys are needed to decrypt the data. In particular, in case of disaster recovery the password-protected key together with the other cloud

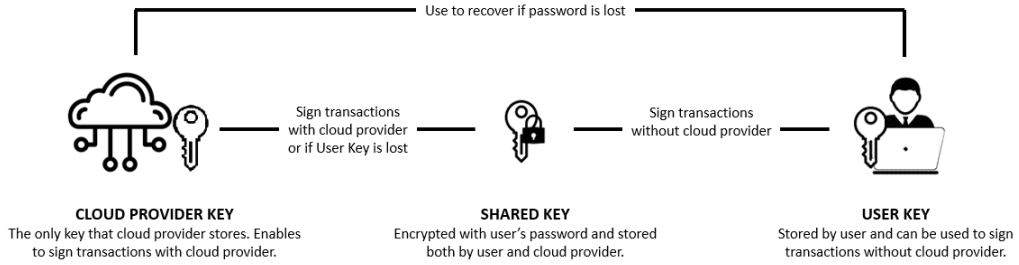

Figure 1: Adopted multisignature schema.

provider key can be used, whereas the two keys held by the company can be used also for offline decryption. Another combination of the two non password protected keys can be used to restore the shared key when the password is lost. This schema has been proposed and adopted also by Coinbase, a digital currency exchange, and has been designed in a way that whichever key is lost, the access to the wallet is not compromised. We started from this scheme and adapted it to the cloud environment.

4.2 Architecture

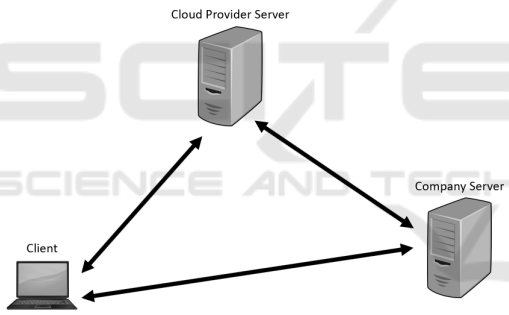

Figure 2: The CoProtect system. Client interacts with Cloud Provider and Company servers and encrypts/decrypts data files.

With reference to figure 2, the CoProtect system is composed of three different entities:

- **CoProtect Client:** a *CoProtect client* (or *client* for short) offers an interface that bridges the data source (e.g., filesystem) and the cloud. It applies encryption (decryption) to the data files and it uploads them to (downloads them from) the cloud. It also interacts with the Cloud Provider and Company servers to perform the necessary cryptographic key operations. For instance, the client software can run on a company employee laptop.
- **Cloud Provider Server:** this server holds one key and a password protected key. In standard operation it is in charge of handling the data decryption procedure or can operate for disaster recovery. It may implement access control policies.
- **Company Server:** this server belongs to the cloud provider's customer and holds a key and a password protected key. It must provide partial decryption service on demand.

4.3 Data Structure

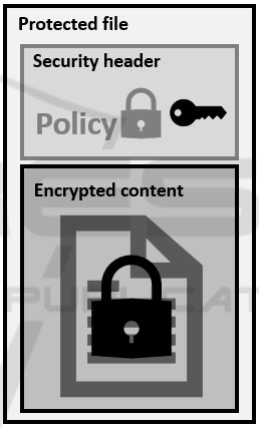

Figure 3: Protected file composition.

To speed-up encryption and decryption processes and minimize the bandwidth needs, we resort to the following encrypting procedure. Every file is encrypted through a symmetric encryption (e.g. using AES) using a random key. This key is in turn encrypted with the public key and appended to each file as a security header, as represented in figure 3. This "enveloping" solution has been used also by Microsoft Azure Information Protection. More specifically, in our case the security header contains the ElGamal encrypted version of a random integer m. Because of ElGamal encryption, this is stored with a couple of values (c_1, c_2) as described in section 3. Given m, it is possible to recover the AES symmetric key by applying the hash function SHA256(m). In the header there can be also policies to regulate data access on a per file basis, such as in case of Microsoft Azure Information Protection, for instance to prevent groups or single persons from

viewing, editing or also printing some kind of documents.

4.4 Operations

We used the Pedersen's algorithm to generate the three keys, ElGamal for secure header encryption and Shamir's Secret Sharing for the distributed decryption. The relevant adopted techniques have been reviewed and summarized in section 3. It's important to note that, in all the operations (key generation, encryption and decryption), the keys are not disclosed to other parties at any time.

4.4.1 Key Generation

The key generation follows the Pedersen distributed key generation summarized in section 3 with three key shares, where two shares are hold by the company and one by the cloud provider. Then the company encrypts one key share with a password and passes it to the cloud provider. The public key is obtained according to equation 1. This procedure must be done once to bootstrap the operations as represented in figure 4.

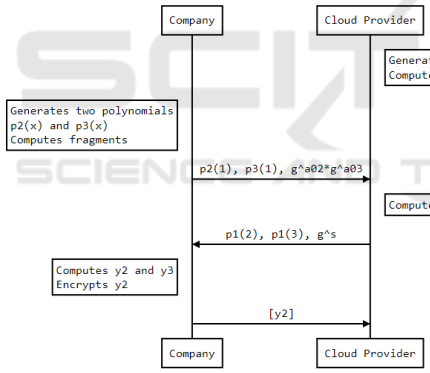

Figure 4: Share exchange for public and private key generation using the Pedersen algorithm.

Data Encryption. Each file is encrypted by the client (owner) with a symmetric encryption with a randomly generated key. This key is then encrypted with the public key and prepended to the encrypted file as an additional header as described in subsection 4.3. The symmetric random key is generated starting from a random integer m extracted in the range $[1, p-1]$ where p is a strong prime described in section 3. Specifically, we apply a hash function on m (e.g. SHA256) to generate 32 bytes in order to create the key for AES256.

Data Decryption. The following procedure happens every time the client wants to access a protected

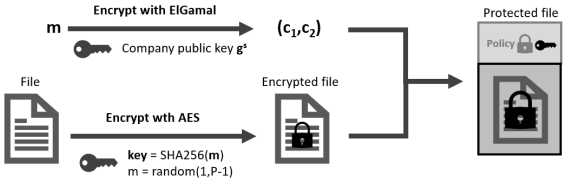

Figure 5: File encryption.

data file:

1. The client takes the file header and sends it to the Cloud Provider Server to have a partial decryption. The result is returned to the client;
2. The client then sends the partially decrypted header to the Company Server that performs a second partial decryption, leading to the full disclosure of the clear-text header file. The result is sent back to the client in a secure way (e.g. using the client public key, or using a secure data channel);
3. The client generates the symmetric key by hashing the number contained in the header file. With the resulting value, it can decrypt the file content using some (relatively) lightweight symmetric encryption system such as AES.

Figure 6 shows the described decryption procedure.

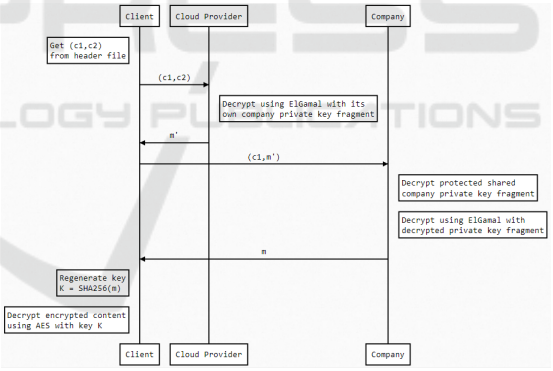

Figure 6: File decryption.

5 PROOF OF CONCEPT

We implemented the described architecture on a proof of concept, whose code is available online (https://github.com/netgroup/coprotect). We developed the architecture elements on a container basis using Docker to implement the client, the company server and the cloud provider server. We implemented the cryptographic API using the python crypto library PyCryptodome for the hashing function and AES. We implemented the Pedersen distributed key generation and ElGamal encryption from scratch. We added a set

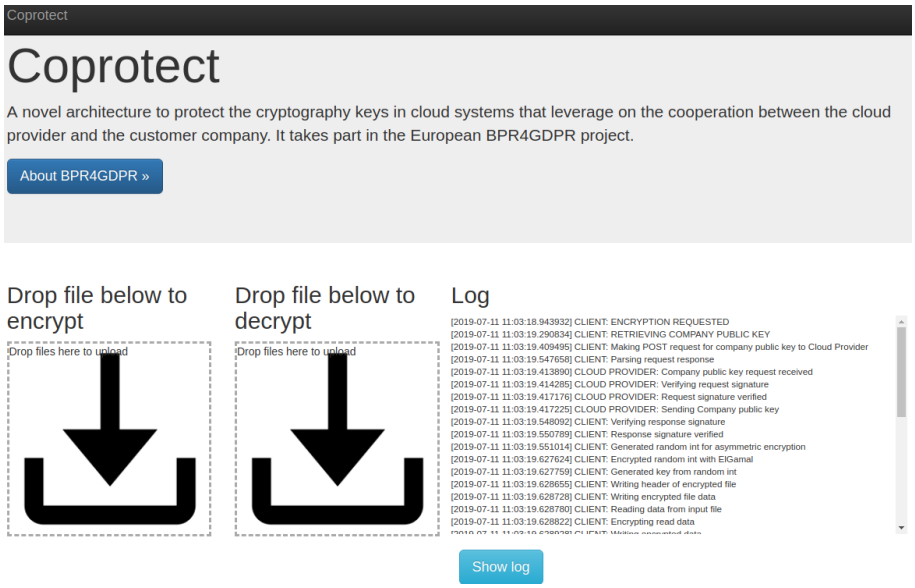

Figure 7: Screenshot of the application user interface.

of rest APIs on all the containers, provided through a Flask application server. The exposed APIs are shown in figure 8 and offer methods to encrypt and decrypt information, getting the log file and the company public key.

We also implemented a user interface reported in figure 7 for testing and demonstration purposes. The interface allows the drag-and-drop of respectively plain-text/cipher-text document for the encryption/decryption.

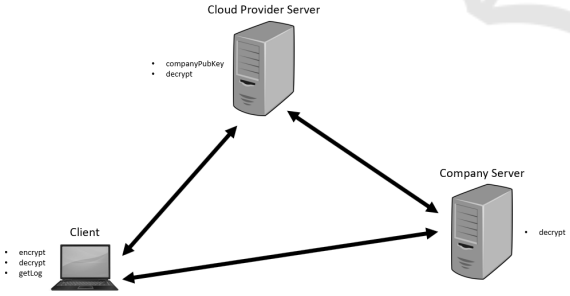

Figure 8: API list.

6 CONCLUSION

In this work we present CoProtect, a system to protect companies digital assets inside the cloud. Differently to conventional systems which entrust just a single entity, the handling of the data is demanded to the collaboration of the cloud provider and the customer companies. The system envisages a fully company control on the access of its digital assets, while offering protection in case of key theft or loss. We implemented the underlying cryptographic system based on Pedersen Distributed Key Generation, ElGamal encryption and Shamir Secret Sharing. We released a proof of concept as a reference implementation and to demonstrate the feasibility. We deliberately keep the solution minimal in order to foster implementation in fully-fledged cloud systems, as it can coexists with the other already deployed systems such as the data access control.

ACKNOWLEDGEMENTS

This work has been partially funded by the BPR4GDPR project from the European Union's Horizon 2020 research and innovation programme under grant agreement No 787149.

REFERENCES

Alliance, (2019). 451 Alliance. https://www.451alliance.com/. Accessed July 2019.

Ben-Assuli, O. (2015). Electronic health records, adoption, quality of care, legal and privacy issues and their implementation in emergency departments. *Health policy*, 119(3):287–297.

Bethencourt, J., Sahai, A., and Waters, B. (2007). Ciphertext-policy attribute-based encryption. In *2007 IEEE symposium on security and privacy (SP'07)*, pages 321–334. IEEE.

Burkhart, M., Strasser, M., Many, D., and Dimitropoulos, X. (2010). Sepia: Privacy-preserving aggregation of multi-domain network events and statistics. *Network*, 1(101101).

CISPE (2019). CISPE Handbook. https://cispe.cloud/. Accessed July 2019.

Coats, B. and Acharya, S. (2014). Leveraging the cloud for electronic health record access. *Perspectives in health information management*, 11(Winter).

De Capitani di Vimercati, S., Erbacher, R. F., Foresti, S., Jajodia, S., Livraga, G., and Samarati, P. (2013). Encryption and fragmentation for data confidentiality in the cloud. In *Foundations of Security Analysis and Design VII*, pages 212–243. Springer.

Ganapathy, S. et al. (2019). A secured storage and privacy-preserving model using crt for providing security on cloud and iot-based applications. *Computer Networks*, 151:181–190.

Huang, Q., Yang, Y., and Shen, M. (2017). Secure and efficient data collaboration with hierarchical attribute-based encryption in cloud computing. *Future Generation Computer Systems*, 72:239–249.

Microsoft (2019). Azure Information Protection. https://docs.microsoft.com/en-us/azure/information-protection/what-is-information-protection. Accessed July 2019.

Nickel, J. (2019). *Mastering Identity and Access Management with Microsoft Azure: Empower users by managing and protecting identities and data*. Packt Publishing Ltd.

Pasquier, T., Bacon, J., Singh, J., and Eyers, D. (2016). Data-centric access control for cloud computing. In *Proceedings of the 21st ACM on Symposium on Access Control Models and Technologies*, pages 81–88. ACM.

Pedersen, T. P. (1991). A threshold cryptosystem without a trusted party. In *Workshop on the Theory and Application of of Cryptographic Techniques*, pages 522–526. Springer.

Puthal, D., Wu, X., Nepal, S., Ranjan, R., and Chen, J. (2017). Seen: A selective encryption method to ensure confidentiality for big sensing data streams. *IEEE Transactions on Big Data*.

Rios, E., Iturbe, E., Larrucea, X., Rak, M., Mallouli, W., Dominiak, J., Muntés, V., Matthews, P., and Gonzalez, L. (2019). Service level agreement-based gdpr compliance and security assurance in (multi) cloud-based systems. *IET Software*.

Rittinghouse, J. W. and Ransome, J. F. (2017). *Cloud computing: implementation, management, and security*. CRC press.

Rushe, D. (2019). Cryptocurrency investors locked out of $190m after exchange founder dies. *The Guardian*.

Sajid, A. and Abbas, H. (2016). Data privacy in cloud-assisted healthcare systems: state of the art and future challenges. *Journal of medical systems*, 40(6):155.

Singh, A. and Chatterjee, K. (2017). Cloud security issues and challenges: A survey. *Journal of Network and Computer Applications*, 79:88–115.

Singh, K. and Batten, L. (2017). Aggregating privatized medical data for secure querying applications. *Future Generation Computer Systems*, 72:250–263.

Song, D., Shi, E., Fischer, I., and Shankar, U. (2012). Cloud data protection for the masses. *Computer*, 45(1):39–45.

Sood, S. K. (2012). A combined approach to ensure data security in cloud computing. *Journal of Network and Computer Applications*, 35(6):1831–1838.

Subashini, S. and Kavitha, V. (2011). A survey on security issues in service delivery models of cloud computing. *Journal of network and computer applications*, 34(1):1–11.

Vu, Q. H., Colombo, M., Asal, R., Sajjad, A., El-Moussa, F. A., and Dimitrakos, T. (2015). Secure cloud storage: a framework for data protection as a service in the multi-cloud environment. In *2015 IEEE Conference on Communications and Network Security (CNS)*, pages 638–642. IEEE.

Xie, H., Yang, Y. R., Krishnamurthy, A., Liu, Y. G., and Silberschatz, A. (2008). P4p: Provider portal for applications. In *ACM SIGCOMM Computer Communication Review*, volume 38, pages 351–362. ACM.

Yüksel, B., Küpçü, A., and Özkasap, Ö. (2017). Research issues for privacy and security of electronic health services. *Future Generation Computer Systems*, 68:1–13.

Zuo, C., Shao, J., Liu, J. K., Wei, G., and Ling, Y. (2017). Fine-grained two-factor protection mechanism for data sharing in cloud storage. *IEEE Transactions on Information Forensics and Security*, 13(1):186–196.