Information Processing and Management 58 (2021) 102745

Contents lists available at ScienceDirect

Information Processing and Management

journal homepage: www.elsevier.com/locate/infoproman

Privacy-enhancing distributed protocol for data aggregation based on blockchain and homomorphic encryption

Cristina Regueiro [1,*], Iñaki Seco, Santiago de Diego, Oscar Lage, Leire Etxebarria

TECNALIA, Basque Research and Technology Alliance (BRTA), Bizkaia Science and Technology Park, Building 700, E-48160 Derio, Bizkaia, Spain

ARTICLE INFO

Keywords:
Homomorphic encryption
Blockchain
Privacy
Confidentiality
Security
Data aggregation

ABSTRACT

The recent increase in reported incidents of security breaches compromising users' privacy call into question the current centralized model in which third-parties collect and control massive amounts of personal data. Blockchain has demonstrated that trusted and auditable computing is possible using a decentralized network of peers accompanied by a public ledger. Furthermore, Homomorphic Encryption (HE) guarantees confidentiality not only on the computation but also on the transmission, and storage processes. The synergy between Blockchain and HE is rapidly increasing in the computing environment.

This research proposes a privacy-enhancing distributed and secure protocol for data aggregation backboned by Blockchain and HE technologies. Blockchain acts as a distributed ledger which facilitates efficient data aggregation through a Smart Contract. On the top, HE will be used for data encryption allowing private aggregation operations. The theoretical description, potential applications, a suggested implementation and a performance analysis are presented to validate the proposed solution.

1. Introduction

With the huge technological progress based on big data and information, concern about people's privacy is increasing. Although there is no concrete definition of the concept of privacy, different points of view usually consider the respect, autonomy and self-determination of people, and how people should be protected from an unreasonable intrusion into his autonomy, even by governments or private institutions. In the technological information context, privacy can be defined as the right of people to control the use of their personal information, including the collection, processing, storage and use of it (Allen, 2000).

On 24 May 2018 the General Data Protection Regulation (GDPR) (Voigt and von dem Bussche, 2017) of the European Union started to regulate the personal data processing activities inside the European Union's territory and market. Data protection in Europe is rigorous, since it imposes the data owner prior consent and allows the possibility of updating or even deleting the files with personal data, among other guarantees. The United Kingdom has also its own Data Protection Act (DPA) (Jay, 2000) based on the GDPR but with exceptions over criminals' personal data as well as data used for scientific or statistical purposes. South Korea, meanwhile, defined the worldwide strictest law, named PIPA (Personal Information Protection Act) (Ko, Leitner, Kim and Jung, 2016) in 2011, covering any data that can be used to identify a person, including images or videos. It also makes a distinction for incomplete sensitive personal data, such as religion or sexual orientation which could be combined with other available data, infringing on personal rights.

* Corresponding author.
E-mail address: cristina.regueiro@tecnalia.com (C. Regueiro).
[1] URL: www.tecnalia.com (Tecnalia Research and Innovation)

https://doi.org/10.1016/j.ipm.2021.102745
Received 2 March 2021; Received in revised form 13 August 2021; Accepted 31 August 2021
Available online 10 September 2021

Additionally, China defined the CSL (CyberSecurity Law) (Balke, 2018) in 2016, covering all forms of personal data belonging to Chinese citizens and avoiding the abroad storage of personal data. However, Chinese authorities should be informed by data holders if data means prohibited activities. In other words, personal data needs to be scrutinized by the processor, somewhat paradoxical in the context of data privacy. Finally, it should be mentioned that the USA is unique among major countries for not having a unified set of data privacy laws despite the huge quantity of global Internet companies (Google, Facebook etc.), giving them a lot of freedom to set their own privacy and data protection standards. However, the State of California approved the Consumer Privacy Act (CCPA), (Abbott and Coon, 2004) that will take effect in 2020, inspired by the GDPR and making all companies responsible for the safe handling of personal data and requiring users to be informed about how their personal data is used and if that data is compromised in any way. This decision in California will be shortly followed by the rest of the states as the public debate on privacy and security breaches is at its peak. According to the high number of international laws and regulations, data privacy and confidentiality can be nowadays considered essential and worldwide organizations must deal with customers personal information in a secure way, providing not only the high availability, data integrity or non-repudiation previously offered but also the new desired confidentiality and privacy.

Without any doubt, Blockchain (Zheng et al., 2018) is a novel and innovative technology which organizations worldwide have started to use in order to increase information security in today's processes. It enables integrity, availability, non-repudiation, and traceability. However, confidentiality is not included by default in Blockchain, so other techniques such as data encryption for data transmission or storage, are required to meet this requirement. In addition, secret computing has also started to gain relevance as it increases data confidentiality and privacy enabling data scientists to compliantly, securely, and privately compute on distributed data without ever exposing it. By this way, sensitive data is unlocked for machine learning, analytical models or simple computations while meeting privacy, security, and compliance requirements. One of the most useful computations over data is aggregation as it can be useful for several disciplines, from energy consumption to counting votes. In this context, advanced cryptographic principles, which keep data encrypted while it is being processed, are needed. Secure Multiparty Computation (SMPC) (Cramer, Bjerre and Buus, 2015), which distributes a computation across multiple parties where no individual party can see the other parties' data, and Homomorphic Encryption (HE) (Acar, Aksu, Uluagac and Conti, 2018), which enables analytical functions to be run directly on encrypted data while yielding the same encrypted results as if the functions were run on plaintext. Each technology has its more suitable applications. In this research work, HE is considered suitable as it is easy to directly perform the aggregation operation and the aggregated result can remain private for all the participants if needed, increasing the variety of potential applications. In addition, HE provides confidentiality not only on data computation, but also on transmission and storage.

The main contribution of this paper is to propose a privacy-enhancing distributed protocol for data aggregation, backboned by Blockchain and HE technologies, for guaranteeing confidentiality on data transmission, storage and computing.

The paper is organized as follows: Section II starts with a description of the state of the art in Blockchain and HE, the two key technologies to be combined in the suggested solution. Section III theoretically describes the privacy-enhancing distributed protocol for data aggregation, which has been then implemented in Section IV. This section describes the implementation details, including the specific Blockchain technology and HE algorithm. In addition, performance measurements are also presented. Section V describes some use-cases which would highly profit from the aggregation protocol from this research. Finally, Section VI draws the key conclusions and future work from the research.

2. Technical background

This section describes the state of the art on the two key technologies involved in the proposed protocol: Blockchain and HE.

2.1. Blockchain

Blockchain (Zheng et al., 2018) has emerged as a secure distributed ledger where participants collaborate for keeping a shared record of transactions without relying on any third party by means of sharing, validating, and storing all the data transactions. Bitcoin (Nakamoto, 2008), presented by Satoshi Nakamoto and used as a broadly available payment system using peer-to-peer technology, was the first public Blockchain network to appear. Later, Ethereum (Wood, 2014) emerged as a public Blockchain with a new ambitious goal based on enabling distributed computation through Smart Contracts. It shortly became the second most famous Blockchain network in the world. However, although Ethereum goes a step further, it still has the same limitations as Bitcoin in terms of scalability, performance and latency. That is why a storm of different Blockchain technologies, such as Ripple (Schwartz, Youngs and Britto, 2018), has come up for the last years.

In addition to public Blockchain networks where there are no restrictions in the participation, private Blockchains have also started to appear to be used with permissioned participants. In this context, Hyperledger Fabric (Androulaki et al., 2018) is the most famous private Blockchain network, allowing access control policies as well as the creation of private channels which guarantee confidentially among them. Ethereum has also its own private Blockchain version, known as Quorum (Consensys, 2018), which also allows the tokenization of assets natively due to maintaining the essence of Ethereum technology and sharing the same virtual machine, the Ethereum Virtual Machine (EVM), for the execution of the Smart Contracts.

Since Blockchain technology appeared, it has evolved from just cryptocurrency systems to decentralized applications (DApps) based on Smart Contracts providing business logic abstraction and execution on a trusted platform. Smart contracts (Cong and He, 2019) are self-executing and self-enforcing computer contracts governed by the explicit terms and conditions established between two or more parties laid out within them. However, although technically possible, it is not possible for the Smart Contracts to properly deal with massive volumes of data due to performance and costs limitations of the Blockchain.

Although Blockchain is a relatively secure technology which provides interoperability, availability, integrity-preservation, traceability and even secure computation, it cannot guarantee data confidentiality by default, making sensitive data transactions almost impossible and, consequently, reducing the number of potential applications for Blockchain. Nowadays, the use of Blockchain technology is spreading across a wave of industries. It is mainly used for interoperable information sharing without compromising data integrity, for automatic actions execution or for providing trusted historical records in sectors such as, for example, healthcare (Gaynor et al., 2020; Khatri et al., 2021), energy (Mashhour and Moghaddas-Tafreshi, 2009; Wang, Li and Zhan, 2021) or supply chains (Blossey, Eisenhardt and Hahn, 2019). However, Blockchain current applications cannot deal with sensitive information with strict privacy rules and, consequently, an additional security layer would be needed.

Different approaches have already been suggested in the literature for privacy preserving data sharing using Blockchain, such as the use of Off-Chain Blockchain Systems (Miyachi and Mackey, 2021), proxy re-encryption techniques (Zou, Lv and Zhao, 2021) or Attribute Based Encryption (ABE) (Bader et al., 2021). In addition, some Blockchain technologies have also tried to improve its privacy level: CryptoNote (Noether, 2014), which provides a high degree of anonymity through ring signatures to obfuscate messages among a group of nodes; Zk-Snarks (Reitwießner, 2016), a novel form of zero-knowledge cryptography and the improved Zk-Starks (Albrecht et al., 2019), which uses collision resistant hash functions instead of asymmetric encryption to get faster operation. Blockchain with HE, allowing private distributed computation, such as in BeeKeeper (Zhou, Wang, Sun and Lv, 2018). HE is, by far, the technology which provides the highest level of privacy in combination to Blockchain.

2.2. *Homomorphic encryption*

Encryption is the process of converting the original representation of some information (plaintext) into an alternative form (ciphertext) in order to prevent potential interceptors from understanding the content. Encryption assures confidentiality in data transmission and storage. However, most of current encryption schemes require of trusting on a third party to compute data since it is necessary to decrypt the received information before analysing or operating over it. By this way, data becomes visible to that third party and confidentiality is partially lost.

HE (Acar, Aksu, Uluagac and Conti, 2018) is a promising form of encryption in which the encrypted result of certain computations on plaintext can be equally obtained by direct calculation on the ciphertext, preventing from undesired information interception and avoiding confidentiality loss and enabling a third party to apply operations on the ciphertext as there is no need to reveal the original data values. Equally to other encryption schemes, HE also uses an encryption key K_e to encrypt plaintext and allows only the one with the matching decryption key K_d to access its data (both with symmetric and asymmetric keys). However, the main difference is that it adds evaluation capabilities for computing over ciphertext without access to the decryption key and remaining the result encrypted. HE allows the performing of a specific algebraic operation $f_{plain}()$, on the plaintext, equivalent to another (not necessarily the same) algebraic operation on the ciphertext $f_{cipher}()$. Considering a ciphertext $c_0 = Enc\ (K_e,\ m_0)$ where m_0 is the plaintext, HE allows obtaining the same result, for any computational function $f()$, operating directly on the ciphertext than operating on the plaintext and then encrypting the result, $f_{cipher}(c_0) = Enc(K_e, f_{plain}(m_0))$.

In order to create an encryption scheme allowing the homomorphic evaluation of arbitrary functions, it is enough to allow only addition or multiplication operations as they are functionally complete sets (Enderton, 2001). The homomorphism always meets $Enc(K_e, f_{plain}(m_0, m_1)) = f_{cipher}(c_0, c_1)$; when $f_{plain}() = +$, the homomorphism property is in the addition while when if $f_{plain}() = *$, the homomorphism property is in the multiplication.

Homomorphic property was firstly used in the rot-13 (or Caesar) (Sanchez et al., 2016) encryption scheme in 1980, considering the concatenation as the computable function. It was not a secure algorithm, but it introduced the homomorphic property concept in the encryption process. Since then, several new HE algorithms have been developed (Parmar et al., 2014). They can be organized in three main types depending on the types and frequency of mathematical operations that can be performed on the ciphertext:

- *Partially Homomorphic Encryption (PHE)*. It allows only one computational function (mostly addition or multiplication) to be performed an unlimited number of times on the ciphertext. Some examples of PHE are RSA (Rivest, Shamir and Adleman, 1978), El-Gamal (ElGamal, 1985) and Paillier (Paillier, 1999).
- *Somewhat Homomorphic Encryption (SHE)*. It allows several types of operations (either addition or multiplication) that can only be performed a certain number of times until a certain complexity is reached since encryption noise increases with each addition or multiplication of the encrypted data resulting in a unencryptable text. This is the precursor to Fully Homomorphic Encryption. The most famous SHE scheme is Boneh-Goh-Nissim (BGN) (Boneh, Goh and Nissim, 2005).
- *Fully Homomorphic Encryption (FHE)*. It can use any efficiently computable function (such as addition and multiplication, not just one or the other) any number of times. Unlike other forms of HE, it can handle arbitrary computations on the ciphertexts. However, nowadays FHE schemes are not possible to be implemented in practice due to the high computational requirements. Some of the most well-known FHE schemes are: Brakerski-Gentry-Vainkuntanahan (BV) (Brakerski and Vaikuntanathan, 2014), Brakerski-Gentry-Vainkuntanahan (BGV) (Brakerski, Craig and Vaikuntanathan, 2014) and Gentry-Sahai-crypto Waterssystem (GSW) (Gentry, Sahai and Waters, 2013).

Table 1 compares the main features of the most well-known HE algorithms. As it can be seen, FHE algorithms' speed is low, making their implementation inapplicable, while PHE algorithms' versatility is also low, reducing their applicability. SHE algorithms are presented as an intermediate option in terms of speed and versatility.

2.3. Previous blockchain-HE privacy enhancing works

There have been some previous studies focusing on combining Blockchain and HE technologies for guaranteeing high security levels in aggregation processes.

Ghadamyari and Samet (2019) present a privacy-preserving method for statistical analysis of health data leveraging the Blockchain technology and Paillier encryption algorithm to increase the accuracy of data analysis while preserving the privacy of patients. It was a high step compared to previous works as it enjoys the benefits of a distributed solution in terms of higher availability while enhancing data security. The proposed scheme guarantees privacy between the different participants (patients) as they only share encrypted data. However, it does not provide patients privacy with the statistics recipient (for example, a researcher), as patients' data is always encrypted with the recipient public key, limiting the statistics to him. Although this study really increases privacy, it still shows some privacy holes that can be improved with algorithms like the one presented in this work. Similar approaches have been presented in Yu et al. (2018) and Park, Chao, Jeong & Park (2017) for voting purposes.

Wang et al. (2019) present a framework combining HE and a hierarchical Blockchain network for data aggregation in smart grids. Although it increases the data aggregation decentralization, the proposal highly depends on special nodes called "gateways" which centralize the link between the different Blockchain levels. If those nodes were attacked, the complete system would go down. On the contrary, the protocol proposed in this research, which also combines HE with Blockchain, solves this limitation presenting a totally decentralized solution while maintaining the required privacy.

3. Data aggregation protocol

This section theoretically describes the proposed aggregation protocol and introduces some potential applications in which it could be useful.

3.1. High-level overview

This research work aims at defining a protocol based on Blockchain and HE technologies, in order to increase the security level of data aggregation processes. In the suggested solution, Blockchain acts as a distributed ledger which facilitates efficient data aggregation through a Smart Contract, guaranteeing secure computation (the aggregation process implemented in Blockchain cannot be altered) as well as availability, traceability, and integrity-preservation. As it has been mentioned before, Blockchain does not properly deal with massive data; so, the protocol neither does due to performance limitations. On the top, each user will locally use HE for data encryption, ensuring confidentiality on the transmission, storage, and computation processes.

Fig. 1 shows a high-level overview of the required architecture for the proposed protocol. As it has been explained, it is based on a distributed Blockchain network over which the aggregation process is deployed by means of Smart Contracts written in Solidity language executed over the EVM.

Each user taking part in the aggregation process requires a python version 3 client with their specific use-case logic as well as the encryption and decryption functionalities with and without homomorphism features (paillierlib.py and RSAcryptography.py in Fig. 1) and the Blockchain needed operations, including wallet functionality as well as Blockchain transactions generation and sending (web3.py). All the communications among participants will be encrypted to increase the overall security of the solution.

3.2. Theoretical description

There are three different users who participate in the protocol described in Fig. 2.

- *Asker.* He is the user who starts the process asking for an aggregated result (Fig. 2a). He is the only one who gets the aggregated result (Fig. 2f), but with no capacity to determine the individual contributions. He is "A" in Fig. 2.
- *Participants*. They are the users who contribute with their own information to the aggregation process (Fig. 2c). Each one is represented as "P_i" in Fig. 2, with a total number of "N" participants.
- *Operator*. He is the participant directly contacted by the Asker (Fig. 2a). The operator role is not fixed as any participant can be contacted by the Asker, thus becoming an operator. He is responsible for starting the aggregation process with his own data

Table 1
Comparison of main HE algorithms characteristics.

Scheme	Hom. Prop.	Type	Security assumption	Speed	Versatility
RSA	$*$	PHE	Integer factorization	High	Low
Paillier	$\oplus$	PHE	Decisional composite residuosity	High	Low
ElGamal	$*$	PHE	Decisional Diffie-Hellman	High	Low
BGN	$*, \oplus$	SHE	Subgroup decision	High	Medium
BV	$*, \oplus$	FHE	Learning with error	Low	Medium
BGV	$*, \oplus$	FHE	Ring-learning with error	Low	High
GSW	$*, \oplus$	FHE	Learning with error	Low	High

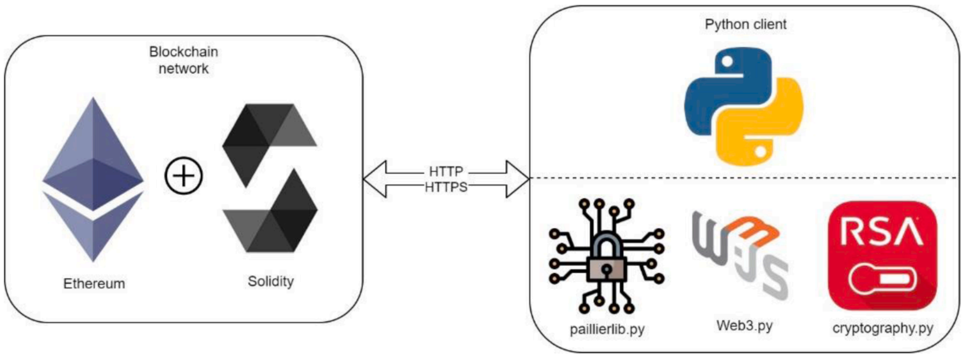

Fig. 1. High-level overview.

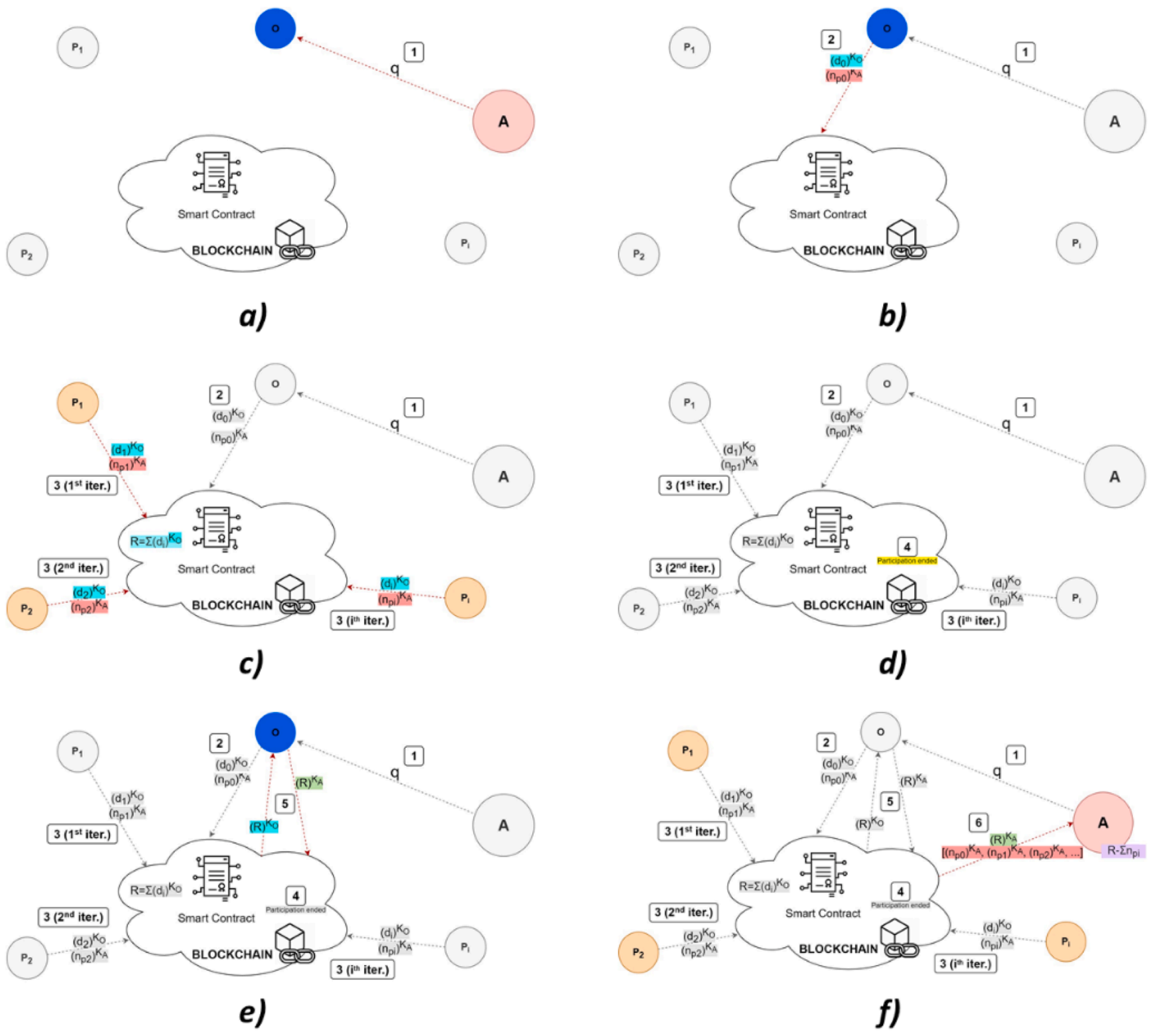

Fig. 2. Protocol diagram.

(Fig. 2b), which will be later enriched with the remaining participants' data (Fig. 2c). When all the required participants have aggregated their own data (Fig. 2d), the operator will send the aggregated result back to the Asker (Fig. 2e). He is "O" in Fig. 2.

The protocol described in Fig. 2 carries out two simultaneous operations. The first one is the private aggregation procedure started by “O” (in blue in Fig. 2b), enriched by “P_i”s (in blue in Fig. 2c) and finally received by “O” (in blue in Fig. 2e). For this purpose, HE encryption (paillierlib.py module from Fig. 1) will be considered using the “O” HE public key K_0, so that he is the only one who can obtain the aggregated result R. However, during the aggregation process, “P_i”s obfuscate their own values with a random nonce so that no one, neither the operator, can really know the real aggregation result nor the individual values. The obfuscated aggregated result will be then shared with “A” using his public key K_A as he is the final recipient (in green in Fig. 2e). This process requires of a second simultaneous operation based on the sharing of each “P_i” nonce with “A” (in red in Fig. 2b, c and f). For this purpose, “A” public key K_A is used (RSA encryption.py module from Fig. 1, as homomorphism is not required in this case, as only transmission and storage must be private) so that he is the only one who will finally know every “P_i” nonce (in red in Fig. 2f) and eliminate the nonces from the obfuscated aggregated result received from “O” obtaining the real aggregated value (in purple in Fig. 2f). Encryption processes are represented as super-indexes while sub-indexes refer to the specific user participation.

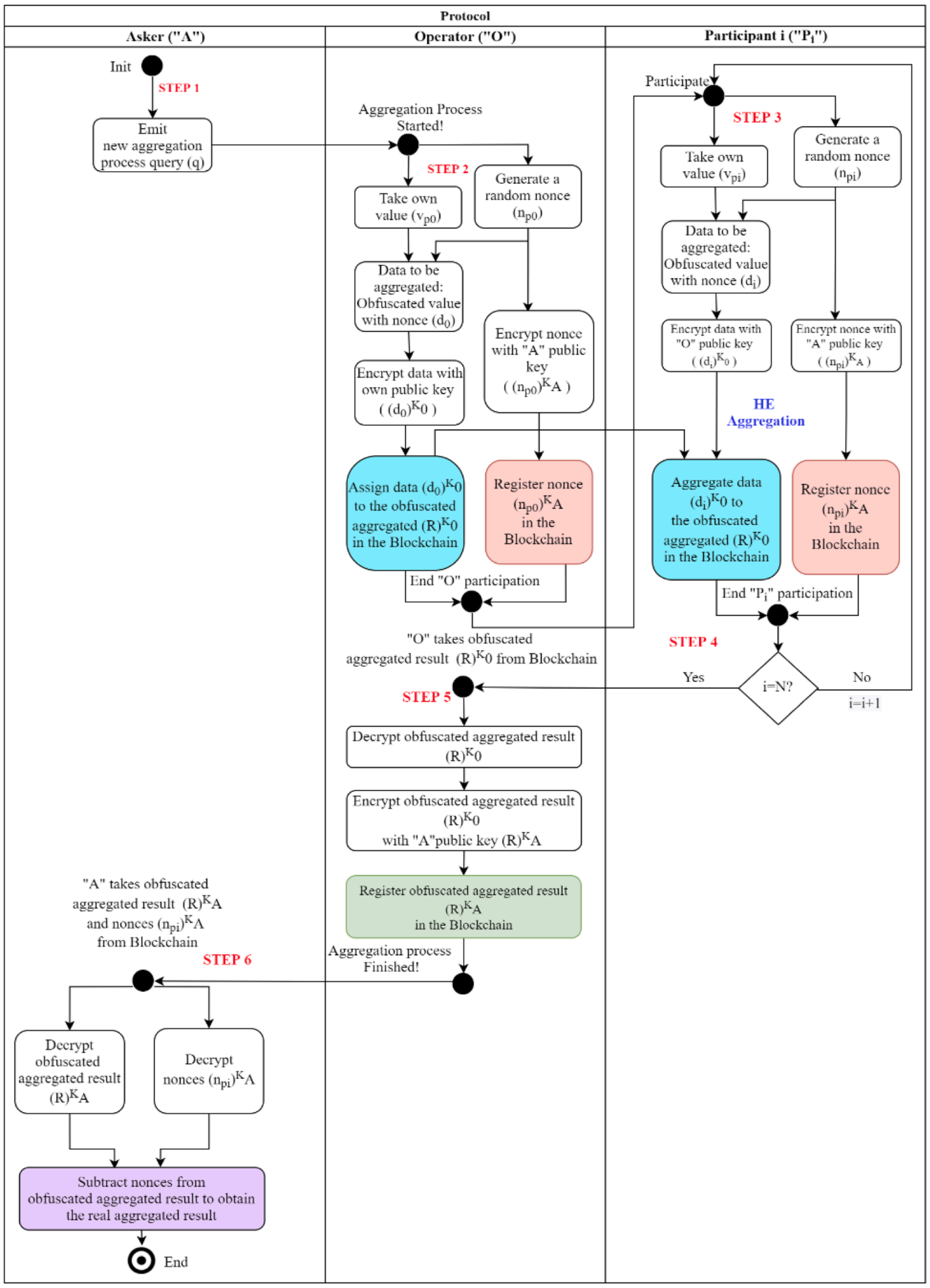
Protocol
Asker ("A")
Operator ("O")
Participant i ("Pi")
Init
STEP 1
Emit new aggregation process query (q)
Aggregation Process Started!
STEP 2
Take own value (vp0)
Generate a random nonce (np0)
Data to be aggregated: Obfuscated value with nonce (d0)
Encrypt nonce with "A" public key ((np0)KA)
Encrypt data with own public key ((d0)K0)
Assign data (d0)K0 to the obfuscated aggregated (R)K0 in the Blockchain
Register nonce (np0)KA in the Blockchain
End "O" participation
"O" takes obfuscated aggregated result (R)K0 from Blockchain
Participate
STEP 3
Take own value (vpi)
Generate a random nonce (npi)
Data to be aggregated: Obfuscated value with nonce (di)
Encrypt data with "O" public key ((di)K0)
Encrypt nonce with "A" public key ((npi)KA)
HE Aggregation
Aggregate data (di)K0 to the obfuscated aggregated (R)K0 in the Blockchain
Register nonce (npi)KA in the Blockchain
End "Pi" participation
STEP 4
Yes
i=N?
No
i=i+1
STEP 5
Decrypt obfuscated aggregated result (R)K0
Encrypt obfuscated aggregated result (R)K0 with "A"public key (R)KA
Register obfuscated aggregated result (R)KA in the Blockchain
Aggregation process Finished!
"A" takes obfuscated aggregated result (R)KA and nonces (npi)KA from Blockchain
STEP 6
Decrypt obfuscated aggregated result (R)KA
Decrypt nonces (npi)KA
Subtract nonces from obfuscated aggregated result to obtain the real aggregated result
End

Fig. 3. Process flow diagram.

The protocol detailed operation, described in Fig. 3, follows the next steps:

1 "A" requires a specific aggregated result and contacts at least one "O" using a query q with the particular use case information, such as, required data, asker public key, finishing condition (timeout for the aggregation, maximum number of participants, participant lists…) etc. The contact between "A" and "O" will be carried out by different means depending on the specific use case: peer-to-peer, broadcasting, transaction on Blockchain etc.
2 Once "O" has received the query, a new aggregation process starts. If several "O"s are contacted, parallel aggregation processes will start.
  2.1 "O" takes his own value v_{p_0} and obfuscate it with a random nonce $n_{p_0}$to obtain the data to be aggregated. $d_0 = v_{p_0} + n_{p_0}$.
  2.2 "O" encrypts d_0 with his own HE public key K_0, to obtain the encrypted data $(d_0)^{K_0}$. He assigns the HE encrypted data to the HE obfuscated aggregated result $(R)^{K_0}$, registered in the Smart Contract $(R)^{K_0} = (d_0)^{K_0}$.
  2.3 In parallel, "O" encrypts the nonce $n_0$with the public key of "A", K_A (HE is not needed in this case, as only transmission and storage must be private), to obtain the encrypted nonce $(n_{p_0})^{K_A}$. The nonce can only be accessed by "A".
3 Once "O" starts the aggregation process, every "P$_i$" can contribute with his own value following a process like the one from "O". This process is repeated for each participant.
3.4 "Pi" takes his own value v_{p_i} and obfuscate it with a random nonce n_{p_i} to obtain the data to be aggregated. $d_i = v_{p_i} + n_{p_i}$, where $i \geq 1$ refers to the specific participant ($i = 0$ is for the operator).
3.5 "Pi" encrypts d_i with the "O" HE public key K_0, to obtain the encrypted data $(d_i)^{K_0}$ to be aggregated to the HE obfuscated aggregated result $(R)^{K_0}$, registered in the Smart Contract $(R)^{K_0} += (d_i)^{K_0}$.
3.6 In parallel, "Pi" encrypts the nonce n_{p_i} with the public key of "A" K_A, to obtain the encrypted nonce $(n_{p_i})^{K_A}$. The nonce can only be accessed by "A".
7 Contributions from "P$_i$" will end when the finishing condition defined in the query is reached.
8 Once the required "P$_i$"s have participated, "O" takes $(R)^{K_0}$ and decrypts it, obtaining the obfuscated aggregated result, including both values and nonces, which is meaningless for "O". He will then share the obfuscated aggregated result with "A" by encrypting it with his public key K_A, obtaining $(R)^{K_A}$.
9 Finally, "A" will get and decrypt the obfuscated aggregated result $(R)^{K_A}$ and every participant nonces $(n_{p_i})^{K_A}$ (including the one from the operator $(n_{p_0})^{K_A}$), registered in Blockchain. With both information, "A" can obtain the real aggregated data by means of subtracting nonce values to the obfuscated aggregated result, obtaining the real aggregated result $R - \sum_{i=0}^{N} n_{p_i}$ where N is the total number of participants in the aggregations process.

3.3. Privacy and security considerations

The suggested privacy-enhancement protocol from this study guarantees the user privacy within the aggregation process thanks to the use of HE, which provides confidentiality to the data transmission, storage, and computation. In addition, anonymity in the participation is, by default, guaranteed as it is not necessary to provide any personal information for using the protocol; an ephemeral (one time-use) Blockchain address is enough (the use of Hierarchical Deterministic wallets (Di Luzio, Francati and Ateniese, 2020) is highly recommended).

Although Blockchain and HE are widely considered secure technologies, there are several studies which point out their main security gaps (Li et al., 2020; Halpin and Piekarska, 2017; Martins, Sousa and Mariano, 2017). This distributed protocol, backboned on these two technologies, will consequently suffer from the same security limitations. Furthermore, there are also some security threats due to the protocol specific operation that could affect the accuracy and truthfulness of the aggregation process. Some of these threats are:

- *False "P$_i$" Injections Attack.* Only authorized "P$_i$"s should take part in the protocol for data aggregation since a non-authorized "P$_i$" participating in the protocol could corrupt the entire system. A countermeasure to tackle this problem could be to make use of whitelists of authorized addresses. The authorized participants will be defined according to the specific use case requirements.
- *"P$_i$" re-participation.* As in the previous case, a multiple participation from the same "P$_i$" could corrupt the entire system. For this reason, it is necessary to limit the number of participations from each "P$_i$" in a single aggregation process. In this case, the addresses which have participated could be registered to be able to verify if some of them has already taken part in the system and avoid its re-participation.
- *False "P$_i$" Contribution.* The aggregation process could be manipulated with false source values from each "P$_i$". It is not possible to avoid false values to be issued but it is possible to achieve non-repudiation by signing each data and nonce provision, making it possible to identify the false data generator. This signing operation could be made in a Trusted Execution Environment (TEE) (Sabt, Achemlal and Bouabdallah, 2015) storing the signing keys in a secure element in the device, to prevent them from being stolen.
- *"A" and "O" possible collusion.* A severe security failure would happen in case "A" and "O" know or share their private keys, since they could decrypt the specific information from all the participants in the aggregation process. It is not possible to avoid "A" and "O" conspiring together to obtain the information from each participant. However, the probability of suffering from this attack highly depends on the specific use-case as, for example, "O" is sometimes predefined in advanced while in other cases, "O" is directly selected by "A" among the available participants.

- *False Obfuscated Aggregated Result.* If “O” were compromised, the obfuscated aggregated result provided in Step 5 could not be reliable. Consequently, it is recommended to use more than one “O” for each aggregation process to prevent this situation, so that “A” would finally obtain an obfuscated aggregated result from each contacted “O”, which should be consistent with the result from other “O”s. In case of different results, a byzantine fault would have happened since at least one of the “O”s could have been compromised. In this context, Byzantine fault-tolerant algorithms (Castro and Liskov, 1999) could protect from “O”s suffering malicious attacks based on exhibiting arbitrary behavior.

In this case, “A” will contact several “O”s for the same query q (Step 1 and 1bis). The proposed protocol from Step 2 to Step 6 will be then repeated in parallel as many times as “O”s have been contacted. Each protocol repetition *j* will be associated to a specific “O_j” using his respective HE public key K_{0j}. It is important to highlight that each “O_j” will also participate as “P_x” for the other repetitions. Finally, a Step 7 is needed for “A” to compare the real aggregated data from the different contacted “O_j”s in order to increase the result trustworthiness checking if they are equal (as expected under normal operation) or not (byzantine fault could have happened). The proposed protocol for multiple operators in shown in Fig. 4, considering in the example two operators, “O_1” and the new “O_2”, showing the new transactions in color. This solution would highly increase the number of transmissions in a number of “O”s (j) factor, since each “P_i” should provide his obfuscated data and related nonce for each aggregation process started by each “O_j”. In this context, different solutions based on sharing the same nonce for different aggregation processes (as shown in Fig. 4) or making several transmissions at once could be useful, reducing the number of transmissions.

3.4. *Potential applications*

The protocol has been described in a generic way to cover a wide range of applications which require of private data aggregation. However, it must be particularized for each application, specifying the participants, limits and other features needed. Some of the most useful applications are described in this section.

3.4.1. *Pandemic control*

With the recent emergence of rapidly spreading infectious diseases, such as the newly emergent human virus SARS-CoV-2 (COVID-19), it is essential to develop preventing tools in order to reduce the transmission extension and, consequently, the sanitary, social and economic impact on our societies. The only available tools to stop the epidemic are those of classical epidemic control: such as case isolation, contact tracing and quarantine, physical distancing and hygiene measures (Ferretti et al., 2020), but they require of knowing if any contact with an infected person has happened. That is why worldwide scientists are studying different technological solutions in the form of anonymized phone location tracking and contact-tracing applications

Therefore, scientists point out that although manual methods help with the spread reduction of a virus, they are not effective. For this reason, systems that allow agile and automatic notifications of all those who have been in contact with an infected person, regardless of whether they know each other or have simply met in a queue at the supermarket, in the gym, playground, school or public transport, can help to rapidly stop the spread of the virus. In this context, technological solutions in the form of anonymized phone location tracking and contact-tracing applications have become essential (Klonowska and Bindt, 2020). Nowadays, many countries have developed their own initiatives, such as China, South Korea, Israel. However, due to the need for quick tools, privacy has not been considered as priority, infringing privacy rights. There are other types of initiatives that have been conceived with privacy and security

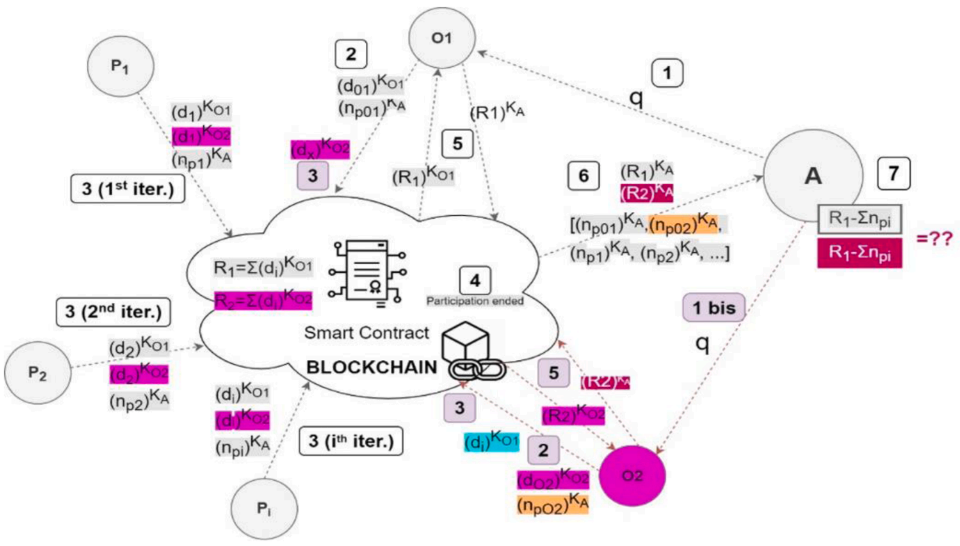

Fig. 4. Protocol with multiple operators diagram.

from design (privacy by design). However, most of them require a centralized server to gather different data sources (location, health etc.), showing several limitations in terms of privacy and security, as it would not be difficult to control users' movements or even re-identify them combining the whole information about a user. In addition, centralized solutions can easily suffer from DoS attacks or false events inclusion. For this reason, decentralized solutions seem to be a suitable solution for maintaining people's privacy and avoiding the violation of citizens' rights. In fact, the DP-3T ("Decentralized Privacy-Preserving Proximity Tracing (DP-3T), GitHub repository 2020) has proposed a couple of designs that aim to provide a decentralized solution that protects users' privacy from design and do not depend on the goodwill and security of centralized services whose information could fall into the wrong hands.

DP-3T solutions are based on informing the user if he has been at risk based on his proximity to infected people. However, the real risk is not specified as the DP-3T solutions only provide Boolean results to the user about having been at risk or not. In this context, the anonymous information about the number of people at risk will also be useful for the sanitary administration in order to forecast potential infections and organize the most suitable response. In this context, the privacy-enhancing protocol for data aggregation presented in this research work seems to be useful, offering a tool that complements other existing pandemic control solutions while preserving citizens' privacy. According to authors opinion, the protocol could be used to count the number of probably infected people in a determined area (area may refer to a country, province, city, town, neighborhood, or even a building) based on the risk status provided by the DP-3T solution. By this way, the real number people having been at risk of infection is undoubtedly known with no need to trust on a third party while patients' privacy is assured.

Let́s consider, for example, a counting application of people at risk of infection for a specific area, such as the Tecnalia building where the authors work in. The application for Tecnalia will have a list of all Tecnalia workers (Tecnalia list) to know when they have finished their participation. At any time, a visitor, such as a client who needs to visit someone in Tecnalia, may want to know how many people in the building are at risk of having been infected in order to determine the real risk of going there. That person (he acts as "A") will ask someone at Tecnalia to provide the required information. The contacted person will automatically start the described protocol (he acts as "O") for anonymously indicating if he has been at risk based on the DP-3T provided results. Besides, he will inform the following person in the list who is "A" and who is "O" in order to make their public keys available. The protocol will afterwards continue the same way following the whole Tecnalia list (they act as "P_i"). Once the list has been completed, the contacted person automatically gets the counting result and sends it to the visitor, who will obtain the number of people in the building at risk of having been infected, being able to determine if going to Tecnalia is secure enough.

3.4.2. *Voting system*

Blockchain advantages are highly desirable for voting applications, as it can be deduced from the huge amount of works researching in this direction, such as Ayed (2017) and Hjálmarsson and Hreiðarsson (2018). Traceability of the whole voting process, as a core requirement, is met by Blockchain as well as anti-tampering and non-repudiation. However, current voting systems based only on Blockchain are not private enough as they are mainly based on the Blockchain anonymization through public and private key pairs (Pawlak and Poniszewska-Marańda, 2021). In this sense, the approach of combining Blockchain with the novel HE technique could help to increase confidentiality.

For this reason, another application for the proposed protocol is to be used as a voting system. In a voting system, it is crucial to guarantee the anonymity of the voter, as well as the secrecy of the vote at the same time as the system can privately count the votes. The voting system could be suitable for several-options surveys as well as true-false voting systems, by coding 0 as false and 1 as true, the final count will give an accurate number of falses and trues.

In this use case, the recipient (it will be "A") will be known while the first voter to participate (it will be "O") will start the vote aggregation process. In this use case, different voters (they will act as "P_i") can issue their own ciphered votes before a specific timeout expires, and the recipient will obtain the total votes without knowing any individual vote.

3.4.3. *Statistics*

Statistics can be useful for a wide variety of disciplines, from physics to social sciences and from health sciences to quality control, as they provide data collection, organization, summary, analysis and interpretation in order to obtain conclusions and take consequent decisions based on such analysis. They try to explain regular conditions in random phenomena.

It is well known that people or organizations are not likely to share sensitive or personal information for just obtaining statistics. Let́s consider, for example, people who suffer from a specific illness. Although statistics in medicine are useful, people may be worry about sharing their personal medical information and make their illnesses public. Something similar happens with working statistics. Let́s consider, for example, statistics about the time a company takes for making a specific product. Making this information public, although is often protected by a legal contract between recipient and the organization, can affect their business if a malicious recipient discloses specific details of their manufacturing processes to the competitors. The supply chain is another context which demands high privacy levels for generating statistics about different material, components, or products (Blossey, Eisenhardt and Hahn, 2019). For that reason, ensuring information anonymity, privacy and confidentiality is essential so that people or organizations are willing to share information to carry out statistics.

Nowadays, statistics are usually done by means of providing individual information and combining it with other individual information in order to anonymize it by just providing the joint result. However, the person/computer/application which receives the individual information is aware of your information (Matthews and Harel, 2011), eliminating confidentiality and making many people not want to participate. In this context, the protocol suggested in this research for data aggregation avoids this lack of confidentiality and could imply an increase in participation in statistics by ensuring the information privacy.

Let́s consider, for example, voting intention statistics periodically carried out by the Centre for Sociological Research or CIS, a

Spanish public research institute. The CIS (he will always act as “A”) would start the survey by contacting to specific well-known trusted participants (they will always act as “O”), for example, one in each province in order to obtain anonymous and confidential information to carry out statistics. Anyone who wants to participate in the survey could provide his voting intention to the trusted participant in his province as well as the nonce to the CIS, as both participants will be always known (and their public keys as well). In this case, the survey will finish when the first 1000 people (for example) to participate provide their vote. The protocol will calculate the aggregated data of voting intention for each political party and will provide the result to the CIS. By this way, every participant has shared a confidential vote which has been considered in the joint information guaranteeing confidentiality and privacy during the whole process.

3.4.4. *Smart grids*

A significant progress in modern power grids is based on the novel capacity of smart meters to provide real-time energy consumption information, being able to aggregate power consumption data from many end-users in one geographical area for intelligent distribution. However, ensuring information security and privacy in the meter data aggregation process is a nontrivial task (Tan, Krishna, Yau and Kalbarczyk, 2013).

Current meter data aggregation systems are mostly based on centralized solutions, in which data from different meters is collected by the energy management system (EMS) (Mashhour and Moghaddas-Tafreshi, 2009), increasing the risks of single point of failure and single point of trust, in other words, the failure and mistrust of the EMS would affect the reliability of data aggregated result. Another challenge of centralized structure is privacy protection as data is firstly decrypted and them summed up to obtain the aggregated value. There is, therefore, a risk that cyber-attackers may eavesdrop the real-time individual meter data by hacking into the receiving nodes.

The smart meters data aggregation application can be useful for several purposes such as, determining the power consumption in, for example, a neighborhood helping the electric utility in various grid-level applications, such as demand-side management, intelligent distribution, load tracking, network expansion, etc (Mattila et al., 2016). A list of smart meters inside the neighborhood should be available to determine all participants. The utility (it will always act as “A”) will periodically require the aggregated data information (“A” public key will be always known by participants as it will be always the same). For the “O” role, there are two options:

1 In the first one, a specific smart meter will always act as “O” and, consequently, its public key will be always known by “P_i”.
2 In the second one, different smart meters take the role of “O” in each repetition. In this case, the security is increased but the “O” public key must be provided.

In both cases, the smart meters participate following the smart meters chain order. Once the chain end has been reached, “O” sends the aggregated data to the utility (“A”), who will know the total power consumption but without knowing any individual information.

4. Suggested implementation

This section describes a suggested implementation of the proposed protocol. The most suitable components in authors’ opinion as well as a performance analysis are presented.

4.1. *Components*

The proposed protocol requires of a HE scheme and a Blockchain network, as they are the two novel technologies combined in the solution. In addition, a Smart Contract and a user application are needed for the protocol operation.

4.1.1. *HE Scheme*

On the one hand, although FHE schemes seem to be the most suitable encryption models for providing different kind of applications, they have been directly rejected as their high computing power requirements are nowadays clearly excessive turning them into not usable. On the other hand, SHE-based models have been considered unsuitable since although different operations are allowed, the number of repetitions for each one is highly limited depending on their complexity. Data aggregation-based applications considered in this study only require addition operations but allowing unlimited repetitions for providing suitable aggregation services for high volumes of data sources. For this reason, PHE is considered the most suitable homomorphic encryption scheme for data aggregation-based services.

Some of the PHE schemes, such as RSA and El Gamal, have a multiplicative homomorphic property while others, such as Paillier, have additive homomorphic property, which is the interesting one for data aggregation-based applications. Besides, although multiplicative homomorphic property can also be transformed into an additive homomorphic property through the Cramer transformation in encryption and a recovery algorithm after decryption, these additional computations require considerable additional computing power as well as time (Cramer, Gennaro and Schoenmakers, 1997). Although Cramer transformation required time is almost negligible in comparison with encryption time, recovery algorithms incur in additional time spending, which is high in comparison to decryption time (Knirsch, Unterweger, Unterrainer and Engel, 2020). Consequently, although it is possible to consider schemes with a multiplicative homomorphic property, which mostly shows better encryption and decryption times (Farah, Javed, Shamim and Nawaz, 2012), algorithms with direct additive homomorphic property are better in terms of total time consumption, especially in the decryption process (decryption time plus recovery algorithm). Within schemes with additive homomorphic property, Paillier is the most well-known for its efficiency as well as its high level of security. Paillier improves other less-known additive schemes, such as

Goldwasser-Micali (GM) and Benaloh (Acar, Aksu, Uluagac and Conti, 2018), in the sense that it can decrease the value of expansion, in other words, a single bit of plain text is encrypted in a lower number of bits, reducing the cyphertext size and, consequently, limiting the required resources and making the scheme really attractive (Sen, 2013).

Any HE scheme is primarily characterized by four operations, which will be specified for the Paillier HE protocol:

- *Key Generation.* It is the operation which generates the required keys (secret/public key pair).

In Paillier, for large primes p and q such that $gcd(p*q,(p-1)*(q-1)) = 1$, compute $n = p*q$ and $\lambda = lcm(p-1, q-1)$. Then, select a random integer g by checking whether $gcd(n, L(g^{\lambda \bmod n^2})) = 1$, where the function $L(u) = (u-1)/n$ for every u from the subgroup $Z^*_{n^2}$, which is a multiplicative subgroup of integers module n^2.

Finally, the public key is the pair (n, g) while the private key is the pair (p, q).

- *Encryption.* It is the classical task of encryption in a conventional scheme.

In Paillier, for each plaintext m, a random number r is chosen for the following encryption operation:

$$c = Enc(m) = g^m * r^n \ (mod\ n^2).$$

- *Decryption.* It is the classical task of decryption in a conventional scheme.

In Paillier, for a ciphertext $< n^2$, the decryption is done by:

$$m = Dec(c) = (L(c^\lambda\ mod\ (n^2))/L(g^\lambda\ mod\ (n^2)))mod\ n.$$

- *Evaluation.* It is an HE-specific operation, which takes ciphertexts as inputs and outputs a ciphertext corresponding to an evaluated plaintext.

In Paillier, the homomorphic property is the addition:

$$Enc(m_1) * Enc(m_2) = (g^{m_1} {r_1}^n (mod\ n^2)) * (g^{m_2} {r_2}^n (mod\ n^2)) = (g^{m_1+m_2} (r_1 * r_2)^n (mod\ n^2)) = Enc(m_1 + m_2).$$

Besides, Paillier has some additional homomorphic properties which describe different cross-relationships between various operations on the ciphertext and the plaintext. However, they are not relevant for this research work.

The Key Generation, Encryption and Decryption operations will be locally carried out by each participant, by means of a web or mobile application, or even a dedicated program. However, Evaluation operation will take place over Blockchain through a Smart Contract.

4.1.2. Blockchain network

Although the proposed protocol can be applied to a wide range of applications, most of them could be clearly oriented to community participation (voting systems, statistics etc.). In this context, public Blockchain networks are more suitable as they enhance transparency and accountability. Other use cases cover the situation in which the access must be controlled, limiting the participants. However, this access control can be implemented within the Smart Contract, maintaining the deployment in a public network.

Despite the huge amount of public Blockchain networks which support Smart Contracts (they are an essential component), Ethereum is the most extended one, guaranteeing decentralization, immutability, non-repudiation and traceability while providing decentralization to the apps built on it (Gencer et al., 2018). It allows running a program in a trusted environment through the Ethereum Virtual Machine (EVM) (Hirai, 2017), guaranteeing it will be equally executed in every peer of the network. Furthermore, Ethereum provides, by default, security in the communication as it is a TCP-based peer-to-peer network, except for the UDP-based

Table 2
Implemented smart contract functions.

Function	Type	Description	Exec	Relevant Param	Step
initAgg	write	Init a new aggregation process and register its aggregation operation	"O"	$(d_i)^{K_0}$ for $i = 0$	2.2
participate	write	Register a new aggregation operation	"P$_i$"	$(d_i)^{K_0}$ for $i \geq 1$	3.2
setNonce	write	Register a new nonce.	"O" "P$_i$"	$(n_{p_i})^{K_A}$ for $i \geq 0$	2.3 3.3
stop	write	Disable participation function.	"O"	-	4
getObfRes	view	Once participation is stopped, take $(R)^{K_0}$.	"O"	-	5
setObfRes	write	Register the aggregated result for "A"	"O"	$(R)^{K_A}$	5
getAggRes	view	Take $(R)^{K_A}$	"A"	-	6
getNonce	view	Take nonce from each participant	"A"	-	6

Kademlia node discovery system (Maymounkov and Mazieres, 2002), as well as high availability and ease of use as users can interact with the network by only owning the Smart Contract specifications and their identification in Ethereum (public/private key pair).

4.1.3. *Smart contract*

Following the protocol description from Section 3.1, the Smart Contract implementation requires the functionalities described in Table 2, where the description, executor, required relevant parameters for the protocol and associated protocol step are included.

The integration of HE and Blockchain technologies happens on the Smart Contract deployed in Ethereum, specifically in the *participate* function where, according to Paillier homomorphic property, a multiplication of two ciphertexts must be implemented to provide the additive property.

$$Enc\ (m_1 + m_2) = Enc(m_1) * \ Enc(m_2) = c_1 * c_2$$

The multiplication operation is optimized in terms of performance by means of the quarter squares (Holmes, 2003):

$$c_1 * c_2 \ = \ \frac{(c_1 + c_2)^2}{4} - \frac{(c_1 - c_2)^2}{4}$$

Although the *participate* function could be used also for "O" in step 2.2, a particular *initAgg* function has been designed in order to improve the performance as the first participation (which is always the one from "O") does not need to be aggregated but only assigned and, consequently, the processing requirements are reduced and the performance improved. In addition, this function also resets all the Smart Contract variables in order to allow its reuse.

Although there are various programming languages for the Smart Contracts development, Solidity is the foremost one since it was contract-oriented developed by Gavin Wood (Antonopoulos and Wood, 2018). It supports inheritance and libraries which, in combination to Web3 collection (Ethereum, 2016), enable an easy way to interact with the applications. In addition, it is a Turing-complete programming language, such as C++ or JavaScript, which can theoretically solve any computational problem provided that there are no other limitations such as memory or gas consumption (maximum amount of work to be done; depending on the difficulty or the speed of a task execution, the computational cost of that operation will be higher or lower and the required gas will increase or decrease in proportion). However, as it is very common to define a gas consumption limit to avoid computational loops or buggy codes in the EVM, the language possibilities are limited in practice.

One of the main drawbacks found in the process of integrating HE with Blockchain is the maximum value to be stored in Solidity, which cannot exceed 256 bits (32 bytes) (Solidity in Depth 2018). Although this size is ideal in Ethereum because it is just the required length for storing hash implementations and addresses, it is not enough when operating with encryption algorithms, such as Paillier. For example, even a simple sum of two 256 bits number may be as wide as 257 bits while the multiplication may be up to 512 bits.

By definition, ciphertext typical bitsize in Paillier encryption is twice as large as bitsize of the encryption key. This size is configurable but a trade-off between performance and security should be reached: the longer the key, the higher the computational cost (higher gas consumption in the Blockchain network) and the higher security level. For security reasons, it is common to consider at least a 2048 bits long key pair, obtaining ciphertexts with a length of 4096 bits regardless of the corresponding plaintext bitsize, expanding the plaintext range to a fixed ciphertext range (Chakraborty, Mathew and Vasilakos, 2018). As it can be easily deduced, ciphertexts required as Smart Contract inputs for the HE have a too long size to be computed in Solidity. For that reason, the BigNumber library (Zcoin, 2019) has been used in this research work to operate with numbers longer than 256 bits. BigNumbers in this library are consecutive 256 bits (32 bytes) words in big-endian order, with the required number of leading zeros such that the value aligns at exactly the size of 256 bits words. As a result, Paillier ciphertexts will be 4096 bits long, in other words, 16 words of 256 bits will be concatenated.

4.1.4. *User application*

Every user will need an application to carry out encryption and decryption local operations as well as to interact with the Blockchain network. It will also include the specific intelligence logic for the specific role and use case. On the one hand, for example, "O" application will provide the intelligence to determine when the possibility to participate must be stopped based on the finishing condition received from "A" on the query, while "P_i" application must look for the "O" and "A" public keys for the aggregation process. For this reason, the user application will be different for each user role. On the other hand, the user application will also differ depending on the use case. For example, in the suggested pandemic control use case, "P_i" application will receive the information about "A" directly from "O" but in the voting system use case, "P_i" application will already know the needed information about "A" since it is fixed.

Due to this high variability on the intelligence logic, a simple user application has been developed in Phyton, importing Web3 for Ethereum Smart Contract interaction and Paillier for HE encryption and decryption processes, but without adding any specific intelligence logic. Additionally, the nonce values encryption and decryption processes, which do not require of homomorphic property, has been carried out with RSA with padding (Rivest, Shamir and Adleman, 1978) due to its wide use.

4.2. *Performance*

Once the protocol has been implemented, some performance tests have been conducted to confirm its applicability in practice. These tests include measuring time consumption, as it factors in how long each operation execution takes, and gas consumption, as it refers to the computational cost necessary to perform a transaction on the Blockchain network. For each test, 10 different iterations have been carried out, taking the average value as representative.

Fig. 5 shows the scheme of the experimental setup following the high-level architecture from Fig. 1.

On the one hand, different platforms with different computing capabilities and memory resources have been considered for the python client in order to measure the influence of these features over the protocol performance and identify if some minimum requirements are needed. The considered platforms are:

- Lat.: Dell Latitude 5590: (1.9 GHz, i7-8650U, 16GB RAM).
- Rpi4-4: Raspberry Pi 4 Model B (Broadcom BCM2711, Quad core Cortex-A72 (ARM v8) 64-bit SoC @ 1.5GHz, 4GB RAM).
- Rpi4-128: Raspberry Pi 4 Model B (Broadcom BCM2711, Quad core Cortex-A72 (ARM v8) 64-bit SoC @ 1.5GHz, 128MB RAM)
- Rpi3: Raspberry Pi 3 Model B (Quad Core 1.2GHz Broadcom BCM2837 64bit CPU, 1GB RAM)

On the other hand, three different Blockchain environments have been considered for the Blockchain network. First, Truffle Suite (Hartel and Staalduinen, 2019), a development environment, testing framework and asset pipeline for Ethereum, which makes possible to locally compile, link and deploy Smart Contracts. It is useful at developing phases to validate the Smart Contract functionality although it does not show a realistic behavior. That is why, to better simulate the real world situation, experiments have been also performed on the two most popular Ethereum testnets, Ropsten (the most Ethereum mainnet similar testnet) (Etherscan, 2021) and Rinkeby (Rinkeby: Network Dashboard, 2021), through INFURA (Consensys, 2021), a gateway which provides Ethereum nodes on the aforementioned testnets.

4.2.1. *Encryption and decryption time*

The required encryption and decryption time for the homomorphic Paillier and non-homomorphic RSA operations have been measured and gathered in Table 3 as they are the two cipher algorithms considered in the proposed protocol.

As it can be seen in Table 3, the required encryption and decryption times for Paillier is at least about 58 times longer for encryption and 7 times longer for decryption than for RSA. This is because Paillier encryption/decryption is a more expensive computing process, especially in the encryption process, in order to provide its homomorphic property and guarantee privacy not only in transmission and storage, as in RSA, but also in the computing process. Therefore, RSA has been considered for encryption and decryption processes not requiring private computing (nonce registering).

These results also show that although Pailler encryption and decryption times are longer than other simpler encryption schemes, they are still suitable for most of the potential applications proposed in this work as it takes no more than 421.86 ms with the lower processing capabilities.

Analyzing the different platforms performance, Lat. is the platform with the best performance, especially for Paillier encryption and decryption processes (around 40 times better performance than Rpi3, the platform with lower processing features); the performance

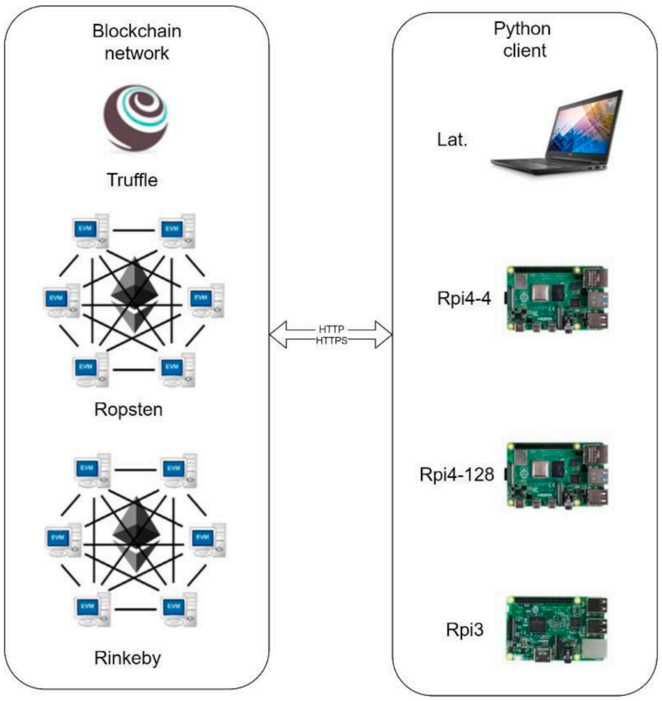

Fig. 5. Experimental setup.

Table 3
Encryption and decryption required average time (ms).

Scheme	Key size	Op	Exec	Data	Step	Time (ms)			
						Lat.	Rpi4-4	Rpi4-128	Rpi3
Paillier	2048	Encr.	“O” “P_i”	d_i for $i \geq 0$	2.2 3.2	10.59	133.13	133.61	421.86
		Decr.	“O”	$(R)^{K_0}$	5	13.17	134.04	133.68	498.28
RSA	2048	Encr.	“O” “P_i”	n_{p_i} for $i \geq 0$ or $(R)^{K_A}$	2.3 3.3 5	0.18	0.52	0.51	1.53
		Decr.	“A”	$(n_{p_i})^{K_A}$ for $i \geq 0$	6	1.77	7.64	7.46	24.12

improvement of Lat. is lower considering Rpi4 platform (around 10-12 times better performance). In RSA, the encryption and decryption time improvement with Lat. is lower, around 3-5 times in comparison to Rpi4 and around 8-13 times in comparison to Rpi3. From these results, it can be concluded that the influence of processing capabilities is much higher for Paillier than for RSA, although platforms with lower processing capabilities are also suitable for Paillier encryption and decryption operations.

Moreover, the influence of the different memory sizes has also been evaluated with Rpi4 (128 MB and 4GB) concluding that it does not really affect the encryption and decryption times.

4.2.2. Blockchain transactions execution time

Table 4 shows the average time needed to perform the functionalities of the protocol which require Blockchain interaction over the three considered Blockchain environments considering an equal gas price of 1 gwei (higher gas price is used for executions getting prioritized). Although the obtained required times on Ropsten and Rinkeby are probabilistic since they highly depend on the current network stability, they are useful to conceive an idea of the expected execution time on public networks.

As it can be seen in Table 4, the Truffle environment shows very low required times as it is a test environment where the block generation is immediate, and no consensus is carried out. Furthermore, Ropsten and Rinkeby show different time results as the Blockchain consensus algorithm they use for the mining process is also different, Ropsten uses Proof of Work (PoW) while Rinkeby uses Proof of Authority (PoA). PoW has a more complex but more secure mining process than PoA. That is why Ropsten requires a longer time for every transaction (almost the double). Although these times could be considered too long for many applications, they are still suitable for the potential applications suggested in this research, which do not have real-time requirements.

Moreover, the average time required for the different protocol steps is also shown in Table 4 for the Lat. platform (the platform with higher processing features). As expected, writing operations consume a longer time as, unlike reading operations, they require a mining process for registering the new transaction. Results in other platforms show higher average necessary times but maintain the same tendency; the required average time increases in about 3 times with Rpi3 (the platform with lower processing features).

Comparing results from Tables 3 and 4, encryption and decryption required average time is almost negligible in comparison with the Blockchain transactions execution, whose time performance mainly depends on the specific Blockchain network as well as the network stability.

4.2.3. Blockchain transactions gas consumption

The average gas consumption needed for the functionalities of the protocol, which require Blockchain interaction over the three considered Blockchain environments, is gathered in Table 5.

As it can be extracted from Table 5, although steps 2 and 3 have almost the same functionality, step 3 is more expensive in terms of gas consumption (almost 5 fives) because it computes a high-consuming aggregation operation while step 2 only computes a lower consuming assignation operation.

In addition, step 2 (and step 3) execution is also more expensive than step 5 because of two reasons: on the one hand, step 2 registers two values (data and nonce) while step 5 only registers one (obfuscated aggregated result). On the other hand, step 2 operates with HE while step 5 uses lighter RSA. Moreover, reading operations, as expected, do not consume any gas.

Finally, it can be seen that the gas consumption for every function execution is almost the same regardless the Blockchain

Table 4
Average time for main transactions (s).

Step	Exec	Smart Contract Functions	Type	Time (s)		
				Ropsten	Rinkeby	Truffle
2	“O”	initAgg + setNonce	write	20.63	14.40	0.52
3	“P_i”	participate + setNonce	write	26.69	14.69	1.13
4	“O”	stop	write	23.26	15.00	0.08
5	“O”	getObjRes	read	0.14	0.14	0.07
5	“O”	setObfRes	write	27.31	14.72	0.18
6	“A”	getAggRes	read	0.16	0.15	0.04
6	“A”	getNonce	read	0.17	0.14	0.04

Table 5
Average gas consumption of main transactions.

Step	Exec	Smart Contract Functions	Type	Gas Ropsten	Rinkeby	Truffle
2	"O"	initAgg + setNonce	write	130 182	130 179	130 181
3	"P_i"	participate + setNonce	write	593 386	593 319	593 117
4	"O"	stop	write	13 538	13 538	13 538
5	"O"	getObjRes	read	0	0	0
5	"O"	setObfRes	write	68 584	68 595	68 590
6	"A"	getAggRes	read	0	0	0
6	"A"	getNonce	read	0	0	0

environment. The reason lies in the gas calculation method in Ethereum, which depends on Smart Contract compiled code, which is the same in the three cases. However, these gas consumption values will grow with the number of participants (the more participants, the higher computational cost and, therefore, the higher gas consumption) until reaching the situation where the required gas consumption exceeds the gas limit defined in the Blockchain and the transaction cannot be properly executed.

4.2.4. Performance considerations

The performance results described in previous subsections show that the proposed privacy-enhancing distributed protocol for data aggregation has certain limitations. On the one hand, there is a limit on the number of participants which depends on:

- The specific computational cost required for the private aggregation process which highly depends on the homomorphic encryption key size; the lower it is, the lower computational cost (lower gas) and the higher the number of participants until reaching the gas limit. However, decreasing the homomorphic encryption key size also means a lower security level, so a trade-off between computation cost and security would be needed.
- The specific gas limit supported by the Blockchain network; the higher the gas limit, the higher the supported computational cost and the higher the number of participants. However, it should not be increased too much to avoid buggy codes or computational loops.

On the other hand, there is a high dependency of the required average time for the protocol execution with the Blockchain network, mainly influenced by:

- The specific computational cost required for the private aggregation process which, as for the gas consumption, highly depends on the homomorphic encryption key size. Higher key sizes mean higher computational cost and, consequently, higher average execution time.
- The Blockchain network stability; with higher network load (higher pending transactions), not every new transaction can be immediately executed and, consequently, higher priced transactions get prioritized.

The described performance results from previous subsections are related to the single operator protocol, which can be considered as reference performance. For multiple operators, the gas consumption can be extrapolated by a number of operators factor while the required average time is not highly modified as although the number of encryption/decryption operations and Blockchain transactions increases in a number of operators factor, these processes are parallel and no much more extra time is needed.

5. Conclusions and open research

This paper designs a privacy-enhancing distributed protocol for data aggregation based on Blockchain and HE technologies which is useful for increasing the security level of a wide range of applications, from statistics to pandemic-control tools. Blockchain offers a distributed ledger to provide integrity, traceability, availability and non-repudiation while HE provides privacy and confidentiality in the data transmission, storage and computation.

The proposed protocol, which has been firstly theoretically described, has then been implemented and evaluated on two testing Ethereum Blockchain networks using Paillier HE scheme and showing promising performance results. More future research can be conducted on applying the proposed protocol to different Blockchain networks and HE schemes as well as implementing and testing the detected security limitations and the suggested countermeasures.

CRediT authorship contribution statement

Cristina Regueiro: Methodology, Formal analysis, Investigation, Writing – original draft, Writing – review & editing, Visualization, Supervision. **Iñaki Seco:** Methodology, Software, Validation, Investigation, Writing – review & editing. **Santiago de Diego:** Methodology, Investigation, Writing – review & editing. **Oscar Lage:** Conceptualization, Investigation, Writing – review & editing, Supervision. **Leire Etxebarria:** Formal analysis, Investigation.

Acknowledgements

This work has been partially supported by the Basque Country Government under the ELKARTEK program, project TRUSTIND (KK-2020/00054). It has also been partially supported by the H2020 TERMINET project (GA 957406).

References

"Decentralized Privacy-Preserving Proximity Tracing (DP-3T), GitHub repository," 2020. [Online]. Available: https://github.com/DP-3T. [Accessed August 2021].

Abbott, B. A., & Coon, C. T. (2004). Financial privacy and the California experience. *Annual Review of Banking & Financial Law, 23*, 411.

Acar, A., Aksu, H., Uluagac, A. S., & Conti, M. (2018). A survey on homomorphic encryption schemes: Theory and implementation. *International Journal of Computer Applications, 51*(4), 1–35. pp. 79.

Albrecht, M. R., et al. (2019). Algebraic cryptanalysis of STARK-friendly designs: Application to MARVELlous and MiMC. In *International conference on the theory and application of cryptology and information security*.

A. L. Allen, "Privacy-as-Data Control: Conceptual, Practical, and Moral Limits of the Paradigm, Faculty Scholarship at Penn Law. 790," 2000. [Online]. Available: https://scholarship.law.upenn.edu/faculty_scholarship/790. [Accessed August 2021].

Androulaki, E., et al. (2018). Hyperledger fabric: A distributed operating system for permissioned blockchains. In *EuroSys Conference*.

Antonopoulos, A. M., & Wood, G. (2018). *Mastering ethereum: Building smart contracts and DApps*. Sebastopol: Russia: O'Reilly Media, Inc.

Ayed, A. B. (2017). A conceptual secure blockchain-based electronic voting system. *International Journal of Network Security & Its Applications (IJNSA), 9*(3), 1–9.

Bader, L., et al. (2021). Blockchain-based privacy preservation for supply chains supporting lightweight multi-hop information accountability. *Information Processing & Management, 58*(3), 1–40.

Balke, L. (2018). China's new cybersecurity law and U.S-China cybersecurity issues. *Santa Clara Law Review, 58*(1), 137–162.

Blossey, G., Eisenhardt, J., & Hahn, G. J. (2019). Blockchain technology in supply chain management: An application perspective. In *Hawaii international conference on system sciences*.

Boneh, D., Goh, E.-J., & Nissim, K. (2005). Evaluating 2-DNF formulas on ciphertexts. In *Theory of Cryptography Conference*.

Brakerski, Z., Craig, G., & Vaikuntanathan, V. (2014). (Leveled) Fully homomorphic encryption without bootstrapping. *ACM Transactions on Computation Theory (TOCT), 6*(3), 1–36.

Brakerski, Z., & Vaikuntanathan, V. (2014). Efficient fully homomorphic encryption from (Standard) LWE. *SIAM Journal on Computing, 43*(2), 831–871.

Castro, M., & Liskov, B. (1999). Practical byzantine fault tolerance. In *Symposium on Operating Systems Design and Implementation*.

Chakraborty, R. S., Mathew, J., & Vasilakos, A. V. (2018). *Security and fault tolerance in internet of things*. Cham, Switzerland: Springer.

Cong, L. W., & He, Z. (2019). Blockchain disruption and smart contracts. *The Review of Financial Studies, 32*(5), 1754–1797.

Consensys. (2021). *Infura: The world's most powerful blockchain development suite*. https: //www.infura.io [Online]. Available: [Accessed August 2021].

Consensys. (2018). *Documentation assets for Quorum* [Online]. Available: https://github.com/ConsenSys/quorum-docs [Accessed August 2021].

Cramer, R., Bjerre, I., & Buus, J. (2015). *Secure multiparty computation and secret sharing*. New York, USA: Cambridge University Press.

Cramer, R., Gennaro, R., & Schoenmakers, B. (1997). A secure and optimally efficient multi-authority election scheme. *European Transactions on Telecommunications, 8* (5), 481–490.

Di Luzio, A., Francati, D., & Ateniese, G. (2020). Arcula: A secure hierarchical deterministic wallet for multi-asset blockchains. In *International Conference on Cryptology and Network Security*.

ElGamal, T. (1985). A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory, 31*(4), 469–472.

Enderton, H. (2001). *A mathematical introduction to logic, 2nd ed.* Boston, USA: Harcourt/Academic Press.

Ethereum. (2016). *web3.js - Ethereum JavaScript API* [Online]. Available: https://web3js.readthedocs.io/en/v1.2.6/ [Accessed August 2021].

Etherscan. (2021). *Ropsten Testnet Explorer* [Online]. Available: https://ropsten.etherscan.io [Accessed August 2021].

Farah, S., Javed, Y., Shamim, A., & Nawaz, T. (2012). An experimental study on performance evaluation of asymmetric encryption algorithms. In *WSEAS European conference of computer science*.

Ferretti, L., et al. (2020). Quantifying SARS-CoV-2 transmission suggests epidemic control with digital contact tracing. *Science, 368*(6491).

Gaynor, M., Tuttle-Newhall, J., Parker, J., Patel, A., & Tang, C. (2020). Adoption of blockchain in health care. *Journal of Medical Internet Research, 22*(9).

Gencer, A. E., Basu, S., Eyal, I., Van Renesse, R., & Sirer, E. G. (2018). Decentralization in bitcoin and ethereum networks. In *International conference on financial cryptography and data security*.

Gentry, C., Sahai, A., & Waters, B. (2013). Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In *Annual cryptology conference*.

Ghadamyari, M., & Samet, S. (2019). Privacy-preserving statistical analysis of health data using paillier homomorphic encryption and permissioned blockchain. In *IEEE International conference on Big Data*.

Halpin, H., & Piekarska, M. (2017). Introduction to security and privacy on the blockchain. In *IEEE European Symposium on Security and Privacy, Workshops*.

P. Hartel and M. v. Staalduinen, "Truffle tests for free - Replaying Ethereum smart contracts for transparency," *arXiv preprint arXiv:1907.09208,*2019.

Hirai, Y. (2017). Defining the ethereum virtual machine for interactive theorem provers. In *International Conference on Financial Cryptography and Data Security*.

Hjálmarsson, F.Þ., & Hreiðarsson, G. K. (2018). Blockchain-based e-voting system. In *IEEE International Conference on Cloud Computing (CLOUD)*.

Holmes, N. (2003). Multiplying with quarter squares. *The Mathematical Gazette, 87*(509), 296–299.

Jay, R. (2000). UK Data Protection Act 1998 - The human rights context. *International Review of Law, Computers & Technology, 14*(3), 385–395.

Khatri, S., et al. (2021). A systematic analysis on blockchain integration with healthcare domain: Scope and challenges. *IEEE Access, 9*, 84666–84687.

Klonowska, K., & Bindt, P. (2020). *The COVID-19 pandemic: Two waves of technological responses in the European Union* [Online]. Available: https://mk0hcssnlsb22xc4fhr7.kinstacdn.com/wp-content/uploads/2021/01/COVID-19-pandemic-technological-responses-EU.pdf [Accessed August 2021].

Knirsch, F., Unterweger, A., Unterrainer, M., & Engel, D. (2020). Comparison of the Paillier and ElGamal cryptosystems for smart grid aggregation protocols. In *International conference on information systems security and privacy*.

Ko, H., Leitner, J., Kim, E., & Jung, J.-G. (2016). Structure and enforcement of data privacy law in South Korea. *International Data Privacy Law, 2*(7), 1–20.

Li, X., Jiang, P., Chen, T., Luo, X., & Wen, Q. (2020). A survey on the security of blockchain systems. *Future Generation Computer Systems, 107*, 841–853.

Martins, P., Sousa, L., & Mariano, A. (2017). A survey on fully homomorphic encryption: An engineering perspective. *ACM Computing Surveys, 50*(6), 1–33.

Mashhour, E., & Moghaddas-Tafreshi, S. (2009). A review on operation of micro grids and virtual power plants in the power markets. In *International conference on adaptive science & technology (ICAST)*.

Matthews, G. J., & Harel, O. (2011). Data confidentiality: A review of methods for statistical disclosure limitation and methods for assessing privacy. *Statistics Survey, 5*, 1–29.

Mattila, J., et al. (2016). Industrial blockchain platforms: An exercise in use case development in the energy industry. *ETLA Working Papers*. The Research Institute of the Finnish Economy.

Maymounkov, P., & Mazieres, D. (2002). Kademlia: A peer-to-peer information system based on the XOR Metric. *International Workshop on Peer-to-Peer Systems*. Germany: Heidelberg.

Miyachi, K., & Mackey, T. (2021). hOCBS: A privacy-preserving blockchain framework for healthcare data leveraging an on-chain and off-chain system design. *Information Processing & Management, 58*(3).

Nakamoto, S. (2008). Bitcoin: A peer-to-peer electronic cash system. *Decentralized Business Review*, 21260.

Noether, S. (2014). *Review of cryptonote white paper* [Online]. Available: https://downloads.getmonero.org/whitepaper_review.pdf [Accessed August 2021].

Paillier, P. (1999). Public-key cryptosystems based on composite degree residuosity classes. In *International conference on the theory and applications of cryptographic techniques*.

Park, D.-S., Chao, H.-C., Jeong, Y.-S., & Park, J. J. (2017). Decentralized E-voting systems based on the blockchain technology. *Advances in computer science and ubiquitous computing: CSA & CUTE 17*. Singapore: Singapore: Springer.

Parmar, P. V., Padhar, S. B., Patel, S. N., Bhatt, N. I., & Jhaveri, R. H. (2014). Survey of various homomorphic encryption algorithms and schemes. *International Journal of Computer Applications, 91*(8), 26–32.

Pawlak, M., & Poniszewska-Marańda, A. (2021). Trends in blockchain-based electronic voting systems. *Information Processing and Management, 58*(4).

Reitwießner, C. (2016). *zkSNARKs in a Nutshell* [Online]. Available: https://blog.ethereum.org/2016/12/05/zksnarks-in-a-nutshell/ [Accessed August 2021].

"Rinkeby: Network Dashboard," 2021. [Online]. Available: https://www.rinkeby.io. [Accessed August 2021].

Rivest, R. L., Shamir, A., & Adleman, L. (1978). A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM, 21*(2), 120–126.

Sabt, M., Achemlal, M., & Bouabdallah, A. (2015). Trusted execution environment: What it is, and what it is not. In *IEEE International conference on trust, security and privacy in computing and communications*.

Sanchez, J., Correa, R., Buena, H., Arias, S., & Gomez, H. (2016). Encryption techniques: A theoretical overview and future proposal. In *International conference on eDemocracy & eGovernment*.

Schwartz, D., Youngs, N., & Britto, A. (2018). *The ripple protocol consensus algorithm* [Online]. Available: https://ripple.com/files/ripple_consensus_whitepaper.pdf [Accessed August 2021].

Sen, J. (2013). Chapter 1: Homomorphic Encryption - Theory and Application. *Theory and practice of cryptography and network security protocols and technologies*. Rijeka, Croatia: IntechOpen.

"Solidity in Depth," 2018. [Online]. Available: https://solidity.readthedocs.io/en/v0.4.24/types.html#integers. [Accessed August 2021].

Tan, R., Krishna, V. B., Yau, D. K., & Kalbarczyk, Z. (2013). Impact of integrity attacks on real-time pricing in smart grids. In *ACM SIGSAC Conference on computer & communications security*.

Voigt, P., & von dem Bussche, A. (2017). *The EU general data protection regulation (GDPR). A practical guide*. Cham, Switzerland: Springer.

Wang, Q., Li, R., & Zhan, L. (2021). Blockchain technology in the energy sector: From basic research to real world applications. *Computer Science Review, 39*, 1–25.

Wang, Y., Luo, F., Dong, Z., Tong, Z., & Qiao, Y. (2019). Distributed meter data aggregation framework based on Blockchain and homomorphic encryption. *IET Cyber-Physical Systems: Theory & Applications, 4*(1), 30–37.

Wood, G. (2014). *Ethereum: A secure decentralised generalised transaction ledger* [Online]. Available: https://gavwood.com/paper.pdf [Accessed August 2021].

Yu, B., Liu, J., Sakzad, A., Nepal, S., Steinfeld, R., Rimba, P., & Au, M. H. (2018). Platform-independent secure blockchain-based voting system. In *International conference on information security*.

Zcoin. (2019). *Big number library for solidity, GitHub repository* [Online]. Available: https://github.com/zcoinofficial/solidity-BigNumber [Accessed August 2021].

Zheng, Z., Xie, S., Dai, H.-N., Chen, X., & Wang, H. (2018). Blockchain challenges and opportunities: a survey. *International Journal of Web and Grid Services, 14*(4), 352–375.

Zhou, L., Wang, L., Sun, Y., & Lv, P. (2018). BeeKeeper: A blockchain-based iot system with secure storage and homomorphic computation. *IEEE Access, 6*, 43472–43488.

Zou, R., Lv, X., & Zhao, J. (2021). SPChain: Blockchain-based medical data sharing and privacy-preserving eHealth system. *Information Processing & Management, 58*(4), Article 102604.