This article has been accepted for publication in a future issue of IEEE Transactions on Network Science and Engineering Citation information: DOI 10.1109/TNSE.2020.2987919

On the Blockchain-Based Decentralized Data Sharing for Event Based Encryption to Combat Adversarial Attacks

Ronald Doku, Danda B. Rawat and Chunmei Liu

***Abstract*—It is human nature to anticipate events in the future and plan accordingly. As such, the possibility of letting future events trigger the decryption of a message is a form of an encryption mechanism we deem to be significant in today's information age. In this paper, we propose a variant of Attribute-Based Encryption (ABE), called Event-Based Encryption (EBE), that will help avoid adversarial attacks. In EBE, we attempt to decrypt the messages in the future after an event is confirmed. We illustrate how EBE can be employed by presenting a scenario where a will is decrypted when an individual has been confirmed dead. To achieve this, we introduce a decentralized data sharing network powered by the blockchain technology that ensures data undergoes a thorough vetting process before it is accepted to the network. This vetted data is useful in determining the facts needed for the confirmation of the occurrence of events and is also helpful in restricting adversarial attacks. Our approach utilizes multi-authority ABE schemes, Natural Language Processing (NLP) techniques, and a decentralized data-sharing platform to achieve our goal. The main contribution of this research is the enabling of multiple parties, each with thoroughly vetted proprietary data, to collaborate to confirm the occurrence of an event. This event confirmation should trigger the decryption of a message. We highlight the various applications of our approach and illustrate the practical and secure nature of EBE using numerical results.**

***Index Terms*—Decentralized data sharing, blockchain, event based encryption, adversarial attacks.**

I. Introduction

A will is a legal document that allows a person to express their wishes of how they would like to distribute their wealth and property at death. The possibility of writing a will usually raises many unexpected emotions such as the fear of death, the anxiety of possibly creating discord in the family, and the financial difficulties incurred for the formulation of a will by a lawyer. Though we can not ease the fears and anxieties associated with the inevitability of death, or settle the strife the death of a relative might trigger in a family after, we are motivated to use EBE to eliminate the intermediary (lawyer) in a situation such as this. EBE might help in lessening the financial burden of the person writing the will. By this, EBE aims to decrypt a message (will) when an event occurs (death), without the need for an intermediary (lawyer).

Manuscript received Day Month Year.

Authors are with the Data Science & Cybersecurity Center (DSC^2), Department of Electrical Engineering and Computer Science at Howard University, Washington, DC 20059, USA. Corresponding E-mail: danda.rawat@howard.edu.

A. Motivation and Potential Applications

This research aims to propose a reliable approach that can predict the occurrence of an event, which consequently leads to the decryption of an associated message. EBE is a network of multiple entities collaborating to confirm the occurrence of an event. These entities are part of a decentralized network, with the blockchain as its underlying technology, that serves as a repository for validated data. The network is segmented into teams that share similar interests. These teams are known as Interest Groups (IG). A data owner's membership into an IG is guaranteed after the data owner's data has gone through a vetting phase to ensure the data is of relevance. This process verifies the integrity of the data utilized in the decryption process. We highlight a few examples of the various ways in which we believe EBE can be employed, consequently, serving as a motivational factor for the undertaking of this work.

1) Privacy Preserving Future Message Decryption: We introduce an encryption mechanism that only decrypts data when an event is confirmed. Such a mechanism can be helpful in situations such as trust funds and wills. This approach removes intermediaries and presents a decentralized method of encrypting data by ensuring the power to decrypt a message does not reside in the hands of a single entity. Furthermore, by removing a middle-man, we can make such undertakings more accessible and affordable, which in turn can increase its adoption by the general public.

2) Verification of Events: In situations where a person has to prove that an event occurred, EBE provides an avenue where that event can be confirmed to have occurred in a trustworthy manner. EBE relies on trusted data collected from multiple sources, all confirming the occurrence of an event. For example, entities such as hospitals and multiple government agencies can collaborate to confirm the birth of an individual. Another potential use-case is situations involving insurance claims where insurance companies can employ EBE during their investigation process. In this scenario, a claim can only be settled if trusted entities such as the police department, hospital, or even eyewitness accounts collaborate to ascertain the authenticity of a claim through the data they own.

B. Contribution

In this work, we introduce a setting where Alice wishes to send Bob an encrypted message that Bob can only decrypt after an event that Alice has specified has occurred. To achieve this, various entities in our network must work together to help in the decryption phase. As mentioned before, decryption only happens when an event is confirmed to have occurred. For the event confirmation process, we employ the collaboration of appropriate entities in the network for the confirmation of the occurrence of an event through the data they possess. This event confirmation process is divided into two stages. The initial phase requires Alice to provide enough information about the event that needs to be confirmed. The network then checks to see if it has the data required to prove that the event has or will happen. If the required data is present, the network creates an access policy that will be employed to encrypt the message, which it sends to Alice.

For the second stage, Alice must encrypt the message using the access policy derived from the first stage, and then send the newly encrypted message back to the network. The network then anticipates the confirmation of this event. When the event gets confirmed, it then sends the message along with the decryption key to Bob. Bob can now be able to decrypt this message. The contribution of this research is evident in the main objectives: i) Enable multiple parties, each with thoroughly vetted proprietary data, to collaborate to confirm the occurrence of an event which avoids adversarial attacks; ii) The confirmation of an event triggers the decryption of a message which was stored for future events, and iii) The encrypted message, along with the keys, is sent to the recipient after an event has been confirmed.

C. Organization

In this section, we discuss how the paper is organized. Section II discusses the work that has been done, which is similar to this research. Section III talks about the underlying network on which the system is built. It discusses the key technologies and assumptions used to implement the network. Section IV presents the proposed approach of the network, which addresses the stages involved in the encryption; event confirmation, and decryption processes. Section V goes into detail on how the system is implemented. Section VI presents the performance evaluation of our approach, and Section VII is the conclusion.

II. Related Work

In this section, we highlight various works that are similar to our approach. We also discuss ABE schemes and Smart contracts which we believe share similarities with this work.

A. Turing-Complete Blockchain Systems or Smart-Contracts

A smart contract is intended to ensure that an agreement between two parties is honored. An example of such a system is the Ethereum project [1], which aims to facilitate the completion of legitimate transactions without the need for intermediaries. The blockchain technology ensures the execution of smart contracts through a decentralized method of honoring contracts among qualified parties. The similarities between our approach and a smart contract system intersect in the desire to eliminate third-parties for a contract to be honored. However, our approach ventures to solve an entirely different problem that smart contracts do not solve.

B. Attribute Based Encryption Schemes

Functional encryption [2] is a framework for flexible data sharing that ensures that different recipients of data see different portions of data. A classic example of functional encryption is ABE [3], which is associated with access formulas. In ABE, a user determines the set of attributes that can decrypt a message in terms of a formula over attributes. For example, a user can encrypt a message that ensures that only students that attend Howard University can decrypt it. A popular example of an ABE is Ciphertext- Policy Attribute-Based Encryption scheme (CP-ABE) [4]. In CP-ABE, the secret keys are associated with attributes, meaning each attribute possesses its key. The decryption process allows a key to decrypt the ciphertext when a set of attributes satisfy the formula the ciphertext (CT) was encrypted under. For instance, if a CT was encrypted under the formula $(A \wedge B) \vee C$, a key that has the attributes A, C, satisfies the formula and can decrypt the message. Our approach is a variant of ABE. However, the novelty of our system lies in the incorporation of nodes with trusted data needed to collaborate in order to confirm the occurrence of an event. Furthermore, the decryption of a message in our approach is solely dependent on the occurrence of a future event, which in reality may or may not happen. Thus, the decryption of a message in our system is not a certainty.

C. Time Lock Encryption Schemes

We also draw inspiration from an earlier work done by [5], where they attempt to implement a Time-Lapse capsule. Our work differs from [5] in the manner that they try to decrypt a message. Their approach involves the decryption of a message after a certain time has passed. However, we decrypt a message based on the confirmation of an event. The similarities of both approaches rest on the fact that they both rely on future events; however, their approach is solely temporal. On the other hand, we rely on the occurrence of a future event happening, which is not always a certainty. We also employ a decentralized data sharing system to confirm an event, which again, is a novel approach in the encryption world. Other approaches that encrypt messages until a specific time in the future have also been attempted. [6] propose such a system in their paper but rely on a third party time server. Rivest and Shamir [7] proposed a time-lock cryptographic puzzle where a trusted third party is responsible for key generation and distribution.

D. Decentralized Data Sharing

There have been various research works involving the use of a decentralized data-sharing framework. The Enigma project, for instance, [8] is being developed to provide a platform where different parties can co-operate to store and perform computations on data, which is kept secure and private. This project is a peer-to-peer venture aimed to be the first decentralized platform that securely stores data and provides privacy-preserving computation. A modified distributed hash-table is employed for data storage. Cybervein [9] is a decentralized platform that manages complex datasets. These datasets can be traded, interconnected, and transformed into structured knowledge. Their architecture can store large amounts of data. Other research works try to combine machine learning and blockchain. OpenMined [10] is another platform that does secure, privacy-preserving machine learning. In this environment, Machine learning models are trained anonymously and in secure environments. Data is uploaded to the platform, where members anonymously download and train the data. Members get rewarded based on their contribution to improving the performance of the final model. Numer.ai [11] is a machine learning-based competition platform where privately secured financial data is used to predict hedge funds. In [12], they use neural networks and homomorphic encryption for predictions without sacrificing prediction accuracy and data privacy. [13] uses alternative proof of work approach called proof of useful work. This was proposed to reward miners based on their contributions. The goal of this approach is aimed at better resource utilization. Users make data available, and the network is divided into machine learning competitions with committees trying to solve the competition.

Our approach, however, is reliant on a decentralized data sharing framework for the decryption of a message. Furthermore, the data-sharing framework utilized in our approach differs from other such frameworks in the manner in which it vets and accepts data. We describe in detail how our decentralized data-sharing framework works in the next section.

III. The Decentralized Data-Sharing Network

A. Blockchain

The efficient utilization of this service hinges on the successful implementation of a decentralized data-sharing network that houses relevant data needed for the confirmation of an event. [14] propose such a system where the underlying technology behind their approach is the blockchain. The blockchain technology has been around for at least a decade now. The world first took notice of the blockchain when it was employed in the Bitcoin cryptocurrency [15]. The success of its employment in Bitcoin instigated the widespread adoption of the blockchain as it helped solve the infamous double-spending problem in digital currency back in 2009 [16]. Its adoption in Bitcoin ensured the blockchain guaranteed data integrity and validity through a computational process known as mining. Mining involved the process of solving a computationally-intensive cryptographic puzzle known as the proof-of-work (PoW). Through the mining process, a new block gets appended to the current blockchain. The PoW determined a hash value that connects the previous blocks to the newly mined block. Subsequently, the result is then broadcast to other nodes in the network for validation. The new block gets appended if the majority of nodes involved in the PoW process reach a consensus.

The blockchain technology has been adopted in various avenues since its usage in Bitcoin. However, scalability issues remain a challenge that has garnered the attention of researchers in the blockchain domain [17]. An approach devised to solve this predicament is *sharding*. Sharding in simple terms can be explained as dividing a blockchain network into multiple teams [18]. Each team is referred to as a shard. Each shard has its ledger and can validate transactions [19]. By splitting the network in this fashion, the network's efficiency is enhanced. These shards collaborate in parallel to maximize the performance of the network [20]. In [14], they propose a new sharding technique that divides the network into teams they call an Interest Group (IG).

B. Data Vetting and the PoCI

Work in [14] focused on addressing the issue of the scarcity of relevant data by devising a novel consensus mechanism known as the Proof of Common Interest (PoCI). The goal of the PoCI was to sieve out relevant data from irrelevant ones. They proposed a network that employed the PoCI to vet and accept data that is relevant. In this section, we build on this work done in [14], as the service we intend to provide is essentially maintained by such a network. The network is divided into teams known as IGs. IGs are created to store data from nodes that share similar interests or data. The reputation of an IG depends on how relevant the data that its members own is. As such, the data of a potential member (PM) of the IG must be vetted thoroughly before that node can be accepted to join the IG. This vetting process is made possible by the PoCI. The PoCI process ensures that nodes accepted into the IG must share similar interests with the other members of the IG. This is enforced by members of the IG that are randomly selected to perform the PoCI on the data of a PM. The PoCI process involves the selected nodes solving a small computational work (PoCI) where they verify that a PM's data aligns with the interests of the IG. The process works by having a genesis data which acts as a yardstick for similarity reference. Similarity reference in this scenario means the data of PMs must share a certain level (set threshold) of similarity with the genesis data. This genesis data is usually owned by an expert in the area of interest, with this expert usually being the originator or creator of the IG.

In our approach, however, certain established entities do not need to be a part of an IG, as their data do not need to be vetted. These entities are established organizations such as Government agencies, hospitals, etc. Such entities are regarded as Tier 1 entities, and their data is deemed to be of the highest quality. However, other nodes in the network are ranked into Tiers based on their reputation in the network. We delve deeper into how the ranking system works for such nodes later on in the paper.

The PoCI process demands the calculation of a unique hash function known as the MinHash of the PM's data. The MinHash is a unique signature of a node's document that can be used to determine the similarities between two documents. To compute the PoCI, we need to find the similarities between the MinHash of the PMs and the members of the IG randomly selected to participate in the PoCI process (Approvers). Each PM is assigned an Approver. For the first set of PMs to be added to an IG, the approver is the owner of the genesis data. An Approver computes the PoCI by comparing its MinHash with the PM's MinHash. This calculation can be done by counting the number of components present in the two signatures. That gives the similarity score for the comparison of any two documents. If the Approvers confirm that the PM's data passed the PoCI, this indicates the PM has proved that it owns relevant data, and as such, can be part of the IG.

IV. Proposed Approach

Our approach has two stages; the data verification stage and the event confirmation stage. The first stage requires that we verify that we have the necessary data to confirm the occurrence of an event. After that is done, the next phase is the event confirmation stage. For the data verification stage, suppose Alice wants to send a message to Bob that needs to be encrypted until a specified event occurs. Alice initially sends a Client Request (CR), which consists of the conditions that must be satisfied in order for the message to be decrypted. The CR is assigned to a facilitator. The facilitator's job during this stage is to ensure that the network has the data needed to decrypt the message. After this is established, the second stage is the event confirmation stage. During this stage, the facilitator sends and an access policy to Alice. Alice encrypts the data using the access policy and sends it back to the facilitator. The facilitator must now gather the keys that can satisfy the access policy. These keys are sent to the facilitator when the nodes confirm the occurrence of the event. The facilitator then sends the encrypted message along with the decryption key to Bob, who can now decrypt the message.

The data verification stage uses NLP techniques to achieve its goal. The initial step in this problem is considered an information retrieval problem. The goal is to discover the nodes that own appropriate data that could potentially help confirm an event's occurrence. We call these sets of nodes delegates. The next step in the process attempts to determine if the delegates can confirm an event. This stage of the problem is treated as a Question and Answer System (QAS). If the results of this step are not satisfactory, the facilitator sends an error message to Alice detailing how the network does not possess the necessary data to determine the occurrence of that event. In the case of a successful outcome, the result of this stage is the generation of a set of nodes known as the Answer Set (AS). The AS are the nodes that have been selected from the delegates to be part of the confirmation process. The facilitator generates the Facts Access Policy (FAP) based on the nodes in the AS. The FAP is the access policy that determines the set of attributes on which the message is going to be encrypted. The FAP is then sent to Alice for message encryption. After Alice encrypts the message, she sends it back to the facilitator. This signals the commencement of the next stage, which is the event confirmation stage. Each node in the AS is now an authority that is capable of distributing attributes. They send attributes to the facilitator after they have confirmed an event. We describe these processes in detail later on. Below are the terminologies we advise the reader to be familiar with to comprehend the process fully:

- **Client Request (CR)**: A Client Request (Fig 1) constitutes a detailed description of the conditions specifiying the event that needs to happen and the '5 Ws and 1 H'.
- **Facilitator**: Each CR gets assigned a facilitator. The facilitator assumes the 'point' role and is responsible for handling the processes that verify the data and confirm an event.
- **Query**: Queries are derived from the description part of the CR. A query is used by the facilitator to determine the delegates. The facilitator sends a query to each member node of the IG, where the nodes that return suitable documents become delegates.
- **Delegates**: Delegates nodes are the top *k* ranked nodes retrieved from the querying process.
- **Answer Set (AS)**: The final set of delegate nodes that have showed they have the required data to prove the occurrence of an event. They are used in the generation of the FAP.
- **Facts Access Policy (FAP)**: This is the access policy used to encrypt the message.
- **Facts-Checking (FC)**: This is the confirmation of a fact by a delegate node.
- **Event**: An event is what needs to be confirmed.
- **Interest Group (IG)**: The network is segmented into IGs. An IG consists of a set of nodes that share relevant data on a topic of interest.

V. How the Network Works

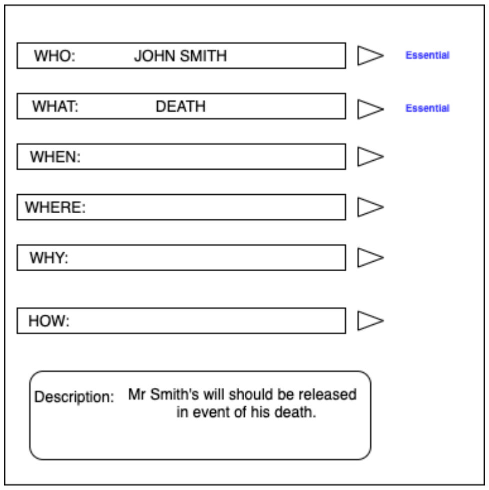
We now proceed to explain how the network achieves the EBE process by highlighting the various processes involved in the data confirmation stage and event confirmation stage.

A. The Data Verification Stage

We begin by describing the processes involved in the data verification stage.

1) Client Request: A user sending a CR to the network (specifying which IG) is the first step in this process. The user fills out a CR which has the '5 Ws and 1 H' segment and the description segment. The '5 Ws and 1 H' fields should be filled out or left blank if that field(s) is non-applicable. Fig 1 illustrates a filled out CR that pertains to a will. In Fig 1, the 'who' field demands the name of the person involved, and 'what' field requests the event that must happen. The other fields can be left blank if no more information is required. However, there is a strict requirement that, at least one field has to be filled out. The answers from the '5 Ws and 1 H' segment of the CR is utilized during the QAS process. The last segment in the CR is the verbal description of the event that needs to happen. The client/user is advised to be as detailed as possible when writing this section, as this part is what is used by the facilitator as a query to find the delegates.

2) Facilitator: The facilitator is the 'point guard' of the EBE process. After an IG receives a CR, the CR is assigned a facilitator. The facilitator takes the description aspect of the CR and turns it into a query. The facilitator sends the query to all the nodes in the IG. The result is a list of top 10 ranked documents called delegates. The next step is the QAS segment, where the facilitator must determine if the delegates possess the knowledge capable of confirming the occurrence of an event based on the answers they give to the '5 Ws and 1 H' questions. The AS (delegate nodes that successfully pass the QAS segment) play a key role in the generation of the



WHO: JOHN SMITH
WHAT: DEATH
WHEN:
WHERE:
WHY:
HOW:
Description: Mr Smith's will should be released in event of his death.

Figure 1. Client Request

FAP that will be used to encrypt the message. After the FAP gets generated, the facilitator sends the FAP to the client. The client then encrypts the message using that FAP and sends it back to the facilitator. The final stage in the process involves the facilitator gathering the attributes from the AS. A node in the AS sends an attribute after it confirms a fact. The facilitator keeps curating these attributes until it gathers the set of attributes that can satisfy the FAP. Once such a set is discovered, this depicts the successful confirmation of an event. The message can now be sent to the recipient along with the attributes encoded in the secret key.

3) Query and Document Ranking: Our goal in this procedure is to measure the similarity between the query sent to the nodes by the facilitator and the documents the data owners own. This process is achieved by using word occurrences and a vector space model that tries to model everything (documents, words, and queries) as a vector in some high dimensional space where individual words are the dimensions. For example, three distinct words have three vector spaces. The bag of words model we employ ignores the order of words in a document, thus documents containing the same words but in a different order end up at the same point on the vector space model. Everything (document or query) is a point in vector space, and our goal is to measure the similarity between two points. To compare documents to queries, we need to determine the distance between these two points in the vector space model. This can be calculated as the euclidean or angle distance between the vectors. The dot product is the building block for all similarity functions in the vector space model. It is the approach used to measure the similarity between two vectors, say A and B, which can be represented as: $a = [a_1, a_2, ...a_d], b = [b_1, b_2, ..., b_d]$ and $a^T b = a_1 b_1 + a_2 b_2 + ...a_d b_d = \sum_i a_i b_i$.

4) Term Weighting: Term Weighting endeavors to find the relative importance of a word (w) in a document (D). D_w represents the coordinate of D along the dimension w. Relating this to queries and documents, the similarity of a query Q to a document D usually takes the form of a dot product. Q_w is the weight of the word w in Q, and D_w is the weight of the word w in the document. The weight of the word can be viewed as the coordinate, which is how far D_w is from the document D on the coordinate that corresponds to the word w. This similarity function can be written as:

$$S(Q,D) = \sum_w Q_w.Dw \tag{1}$$

We, however, need to set weights that lead to a good similarity function. To achieve this, we need to take into consideration the presence or absence of words. A binary vector that sets a word that is present to 1, and 0 if otherwise is utilized. This approach does not take into consideration how many times a word appears. It just sets it to 1. It simply looks out for common words that are between the query and the document without counting repetitions. It is also worth noting that keywords tend to be repeated in documents. Hence, let $tf_{w,D}$ be the number of times w occurred in D. If a word occurs multiple times, the assumption is that it is important. As such, we want the number of occurrences to be a direct reflection of the importance of that word. We call this Term Frequency (tf), and it becomes the weight of the word:

$$S(Q,D) = \sum_w tf_{w,Q}.tf_{w,D} \tag{2}$$

However, this approach tends to be biased towards long documents as they contain more words. Because of this, we have to normalize by document length $|D|$. Thus, for the weight D_w, we use the term frequency of $W_D \times |D|$. This is, however, only done to the document, not the query. The reason for this is because we only have one query and many documents. And these documents vary in size; thus, what matters more is how these documents get ranked because the query always remains constant from document to document.

5) Inverse Document Frequency: Rare words carry more meaning than words that occur too frequently. As such, we would like to make such words important by giving them more weights. Rare words usually carry the content but are also not encountered too often. To determine the meaningful words in a text would mean observing the frequency of all the words in a large body of text. If a word is mentioned once, it could be a 'typo'. However, if it is mentioned three or five times, it might be a very specific word. Inverse Document Frequency (idf) is an approach used to give weight to words. It is thought of as the probability of a randomly picked document containing the word w, which is usually a small number for rare words. To make it more effective, it is done by giving more weight to rare words using: $log\frac{|c|}{df_w}$, where $|c|$ represents the number of documents in the collection and $|df_w|$ is the number of documents containing w. The log is used to put idf on the same scale as the tf component. With this, we can create a new similarity formula:

$$S(Q,D) = \sum_w tf_{w,Q}.\frac{tf_{w,D}}{|D|}.log\frac{|c|}{df_w} \tag{3}$$

6) Frequency Normalization: Documents diminished with high *idf* words usually propagate up to the top if they happen to match the query, which turns out not to be a desirable trait. To remedy this, we make the first occurrence of a word more important than a repeat occurrence. This is because words are contagious. If you see a word once, you are likely to see it

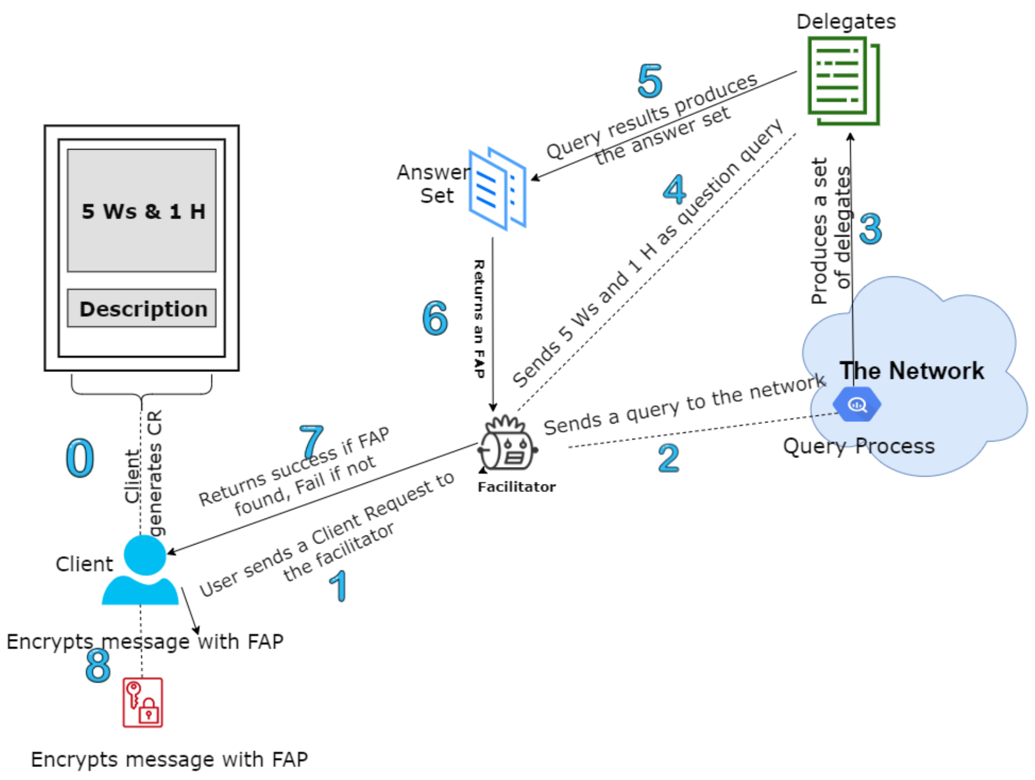

Delegates
5
Query results produces the answer set
Answer Set
4
Sends 5 Ws and 1 H as question query
Produces a set of delegates
3
5 Ws & 1 H
Description
6
Returns an FAP
The Network
Sends a query to the network
Query Process
2
Facilitator
0
Client generates CR
7
Returns success if FAP found, Fail if not
User sends a Client Request to the facilitator
1
Client
Encrypts message with FAP
8
Encrypts message with FAP

Figure 2. The Data Confirmation Process

again. The level of surprise decreases as the word frequency increases. This can be encoded to the ranking formula by using a squash function. This is done to squash the growth of the term frequency. A higher contribution is given to the occurrence of the first word, and a lesser contribution to the second occurrence and more diminishing returns to further occurrences of the word. An effective squashing technique is $\frac{x}{x+k}$ which asymptotes to 1 at high values of term frequencies. k is a meta parameter, and it is used to control how aggressive the squashing needs to be. However, the frequency of terms still needs to be taken into account, especially in long documents. Long documents must have large values of k (squashing factor) because the function needs to grow steadily. To make k dynamic, we take the document length and the term that needs to be ranked. Each document is compared to the average document length in the collection, and if it is a much longer document, then k is adjusted to be big. If it is shorter, the approach is to view it as a step-like function, which is either 1 or 0. This then becomes:

$$\frac{tf_{w,D}}{tf_{w,D}+\frac{k|D|}{avg|D|}} \tag{4}$$

7) Final Formula: Combining all of these together gives a formula that can be used to rank documents against a query. The ranking formula then becomes:

$$S(Q,D)=\sum_{w} tf_{w,Q}\cdot\frac{tf_{w,D}}{tf_{w,D}+\frac{k|D|}{avg|D|}}.log\frac{|c|}{df_w} \tag{5}$$

The facilitator selects the top 10 documents returned by this formula. These documents become the delegates that will participate in the next step.

8) Question and Answer Based System: This section strives to find documents that have data that could possibly answer factoid questions. Factoid questions are questions that can be answered by a simple fact. These questions are usually questions about locations, names, etc. Our approach requires that we find a suitable answer to the ‘5 Ws and 1 H’ questions in the CR by examining the delegate documents. From the ‘5 Ws and 1 H’, we generate a supervised machine learning algorithm that has been trained to formulate questions based on the answers from the CR. For instance, for the scenario in Fig. 1, the question the supervised model generated is ’Is John Smith dead?’. This factoid question incorporated the ‘who’ and ‘what’ answers of the CR to frame a question that needs a specific type of answer. The desired answer should have a named entity of type ‘person’ and an event of type ‘death’. This information would be used to probe the delegate documents for answers. From these delegate documents, we extract passages that will be processed in an answer processor to scan for the right answer. This process has three main steps:

1) Question Processing: Formulates the question and determines the answer type.
2) Passage Retrieval: Break delegate documents into suitable passages.
3) Answer Processing: Extract candidates’ answers and rank candidates.

The initial step is to figure out what the question is about.

This approach determines the answer type we are looking for. For that, we use an answer type taxonomy from Li and Roth [21]. Li and Roth have six coarse classes, which are: Abbreviations, Human, Location, Entity, Description, and Numeric. Inside those 6 coarse classes are more specific classes (50 finer classes). For example, for the coarse class location, the finer classes are city, country, mountain; for human, we have group, individual, title, description as the finer classes; etc. To detect the answer type, we train a machine learning model for various question types and then train them on classifiers using some rich set of features. These features include rules, questions and word phrases, named entities, and semantically related words.

The next step is to segment the documents from the delegate set into shorter units. Paragraphs are appropriate chunks, and a paragraph break is a good 'segmentor'. We then re-rank the passages based on the answer types we are expecting. This passage ranking can be done by using certain features. For example, these features could be how many named entities of the answer type occur in the passage, how many query words occur in the passage (in overall relation to the document). To process the answer, a Named Entity Tagger is run on the passages. For the scenario above, the full answer type we seek should contain the name of a person and the event. However, we should only expect the full answer at the event confirmation stage. For this stage, we believe partial answers would give us a fair indication of the possibility of getting a full answer. For instance,'Jane Doe is dead' is regarded as a partial answer, whereas 'John Smith is dead' is a full answer. This is because the partial answer only contains a fragment of the answer type we seek. However, even though 'Jane Doe' does not match the name we are looking for, we can deduce this delegate node is likely to have the information we are looking for should it happen in the future since it has succeeded in predicting the two named entities that make up the final answer. This because the 'Jane Doe' is a name and 'John Smith'" is a name as well. This matches the semantics of the answer type we desire. After picking the delegates that satisfy these conditions, the next step is to rank them. We count how often a node's document has the desired answer type we want. That is, we take into account how often 'x is dead' is answered by the node, where x is the name of a person. We call this count C. Furthermore, a node's reputation in the network plays a part in the ranking of the answers. We describe in the next section how a node's reputation is calculated. We also use Mean Reciprocal Rank (MRR), which returns a ranked list of M candidate answers for each query. The score of that query is $\frac{1}{Rank}$ of the first right answer or 0 if no right answer is found. We then take the mean of those ranks over all N queries. This then becomes the formula for ranking:

$$MR^3C = \frac{\sum_{i=1}^{N} \frac{1}{rank_i}}{N} \times RC \tag{6}$$

.

We pick the nodes whose score meets a threshold. These nodes are then deemed to have the ability to successfully confirm the occurrence of an event should it happen, and these nodes become the AS.

9) Tiers, Reputation, and Facts Access Policy: To calculate the reputation of the node in the system, we take into account the number of times it has participated in the PoCI process and the number of PoCI processes that happened after the node became a member of an IG. This is essential since we use that to gauge a node's involvement or inactivity in the network. Every node is expected to keep track of any new member that is added, their PoCI activities, and the overall number of PoCIs that have occurred since their arrival. We can calculate the Activity Rate (AR) of members in the network by simply keeping track of the number of times they have participated in the PoCI process since their addition to the IG. To calculate the AR of a node i, we take the number of participations (NP) over the number of PoCI processes (P) which becomes ($AR = \frac{NP}{P}$). All the nodes are expected to have the same score for the node i. To confirm the right answer, we employ a consensus mechanism that chooses an answer that is backed by at least $\frac{2}{3}$ majority of the members. Members are expected to participate in this process to enhance their credibility in the network.

In our network, the likelihood of a node moving up a tier is dependent on its participation in 16 event confirmation processes. This leads to the generation of a relevance vector. The relevance vector is what determines whether a node gets promoted or not. To achieve this, we adopt the approach by [22], where they choose a *k-bit* length vector. In our case, we chose k to be 16 (the steps required) because we conducted experiments that showed 16 delegation processes worked better than the other alternatives (8, 24, and 32). Thus, we have a 16-bit length binary vector where a bit of 1 in the 16-bit sequence denotes a node's data has been selected to be a delegate, and 0 denoting failure to get selected during 16 data verification processes. Attached to each relevance vector is a number m, which represents the number of most significant bits. The m significant bits are found by counting bits to the right. To evaluate the score of a relevance vector, we count up to the m^{th} significant bit and then convert it to an integer. After this is done, we divide it by 2^m. For instance, if we want to get the score for a relevance vector for a node i from node j, with the relevance vector of rv_{ij} = 1110011000000000 where m= 7. The score it represents can be calculated as: $\frac{(1110011)}{2^7} = \frac{115}{128} = 0.8984$. This gives us a value between [0 to 1). Everyone in the IG is expected to calculate this value. We also keep track of a non-relevance vector, which is just the complement of the relevance-vector. We subtract this from the final score of the ranking as a way to deter malicious activities by penalizing falsification. Also, the reward of 0.05 is added to the score of each node that is selected to be part of the AS. This rewards nodes with quality data by expediting their process of moving up a level with extra points.

The next step is the calculation of credibility. A node earns credibility by accurately providing a relevance vector score when the system requests it. A credibility vector is created using the same approach that we used for the relevance vector. Each node is required to have a credibility vector of all the other nodes in the network. To determine the correct credibility score of a node, we follow the $\frac{2}{3}$majority consensus approach. The nodes that do not have this score correct are penalized with a 0, whereas nodes that pass are given a score of 1. Nodes that are not confident in their score and do not wish to be penalized can choose not to participate in providing a credibility score. However, non-participation does not augur well for such nodes as it leads to a low credibility score which prolongs their chances of getting into the next level.

Putting all of these together, the formula to determine the reputation score of a node is:

$$r = \sqrt{AR} \cdot \frac{\sum_{i=1}^{k}(c_i r_i) - n_i}{k} \tag{7}$$

where AR is the activity weight. To diminish the influence AR has on the overall score by giving it less weight, we take its square-root. c_i and r_i are the credibility and relevance vector scores, respectively. We subtract n_i (non-relevance score) from the score as a way to punish falsification.

Nodes are grouped into tiers based on their score and their status. For instance, institutions such as hospitals, and government agencies are deemed to be trustworthy and, as such, automatically fall in the highest Tier (1) level. There are 4 Tiers in all. A node is only legible to move to the next Tier if its score after 16 data verification processes satisfy the threshold. Nodes that have a reputation score of $.80 - 1$ fall in Tier 1 which is the highest, $.60 - .79$ is Tier 2, $.25 - .59$ is Tier 3 and $0 - .24$ is Tier 4, the final tier. Every node in the network maintains a reputation table.

The FAP formula is generated by employing Tiers as attributes. For instance, if users want a resilient access policy for their data, they can choose an access policy that requires a formula $(T_1 \vee T_1) \wedge (T_2 \vee T_2 \vee T_3 \vee T_3)$. This would require two authorities in the Tier 1 level or two authorities in Tier 2 and two authorities in Tier 3 to partake in the key generation phase. Formulating a FAP with those requirements would compel us to know how many nodes in the AS belong to the different levels of Tiers. For instance, if the AS does not have any Tier 1 level node, we can not create a FAP that requires a Tier 1 level node to be part of the boolean formula. On the other hand, if the answer set has four Tier 1 level nodes, an appropriate FAP can either demand one, two, three, or four Tier 1 level nodes depending on how strict the FAP is required to be. In this manner, a strict FAP reflects a high level of confidence in a confirmed event. Thus, the presence of more Tier 1 level nodes in the FAP assures a more confident and secure event confirmation.

After the FAP is decided on, the facilitator sends a success message to the client that the network can confidently handle the request. It also sends multiple FAPs based on difficulty levels (permutations consisting of how many different tier levels should be present) to the client. Higher tier levels in the FAP assures a more confident result. Clients choose the FAP policy they desire and then encrypt the message with it. The encrypted message is then sent back to the facilitator.

B. The Event Confirmation Stage

This section describes the final stage of the EBE process, which is the event confirmation stage.

1) Event Confirmation: Event confirmation occurs when the answer type gets upgraded from a partial to a full answer. For instance, for the example above, the correct answer to 'Is John Smith dead?' is 'John Smith is dead'. In this scenario, our 'who' and 'what' questions are answered completely and not partially. When this happens, every node in the AS that can successfully produce a full answer must send its attribute (T1, T2 , T3 or T4) to the facilitator. The facilitator then checks to see if the attributes it has gathered satisfies the FAP.

2) Attribute Based Encryption: Conventional public-key cryptography requires that a message be encrypted for a particular recipient utilizing the receiver's public key. Identity-based encryption (IBE) transformed the conventional perception of public-key cryptography by allowing the public-key to be an arbitrary string. Consequently, ABE can be explained as a public key encryption mechanism that allows for the encryption and decryption of messages based on the attributes of the receiver. ABE was first introduced by Sahai and Waters, and is also known as Fuzzy-Identity based encryption (a variant of Identity Based Encryption [3]). The classic scenario for the application of ABE was in the area of using fingerprints in Identity Based Encryption. This was because fingerprints have peculiar characteristics that make it unique. For example, a fingerprint may have a set of 40 characteristics and not all of these characteristics may match. Consequentially, this may fail to decrypt a message. The approach to address this was the Fuzzy-Identity Based encryption. In this approach, a threshold is set for d many characteristics. Hence, if there are d or more matches in the fingerprint, then the message should able to be decrypted.

The modern form of ABE, however, pushes the envelope by defining identity as a set of attributes that allows messages to be encrypted to this set of attributes (key-policy ABE [23]) or a policy generated over a specified set of attributes (ciphertext-policy ABE [4]). This approach requires that an entity should only decrypt a ciphertext when the attributes of the keys match the attributes of the ciphertext. In most cases, a central authority issues user keys. In a recent form of ABE [4], private-keys are connected with a set of attributes, and a ciphertext defines an access policy over a specified universe of attributes within the system. A user decrypts a ciphertext if the attributes he/she has collected satisfy the policy of that ciphertext.

In most ABE systems, there is usually a single authority distributing the keys. Keys reflecting attributes are distributed to users in the system by a central authority based on the attributes a user possesses. Because there is one central authority responsible for giving out all the keys, such a system does well to fend off collusion attacks. However, because of its centralized nature, such a system is also susceptible to Denial of Service (DOS) attacks and bottleneck performance issues. Lewko and Waters [24] propose a system to remedy these issues wherein they aim to decentralize ABE. In their approach, an entity encrypts a message for an access policy formulated over attributes given by various authorities. Their principal contribution is that each authority can independently distribute keys on their own without any interference from a central authority.

Our problem can now be relegated to the work done by [24], which tries to decentralize CP-ABE. In our approach, the nodes in the AS become authorities that give out keys that reflect their attributes to the facilitator. The facilitator's penultimate role is to gather the keys that satisfy the FAP. The universe of attributes is defined to be $T1, T2, T3$, and $T4$. A node in the AS only sends out a key reflecting its attribute to the facilitator whenever it finds the full answer that confirms an event. The facilitator must keep collecting the attributes until it attains the attributes that satisfy the access policy chosen by the user. For example, if a ciphertext is encrypted employing the policy $(T2 \vee T3) \wedge T1$, the facilitator must keep collecting attributes until it has $\{T1, T2\}$ or $\{T1, T3\}$.

3) Message Encryption and Decryption Process: The flexibility [24] gives is that any entity can become an authority by generating a public key and distributing private keys to any user that reflect their attributes. In our approach, the facilitator can reflect all the four attributes, and as such, can gather the keys needed to formulate a decryption key. Furthermore, global identifiers also link the private keys that were distributed to the facilitator by the AS nodes [25]. We proceed to describe how the algorithm encrypts and decrypts messages employing the following steps below.

- Global Setup (λ)→GP: This step uses a security parameter λ as input and produces global parameters.
- Authority Setup (GP) →SK,PK: Each authority (a node in the AS) controls its authority setup. The setup algorithm takes the global parameters and the node's characteristic attributes as input. Then, for each attribute that the node controls, the node produces a secret key SK and the corresponding public key PK. The public key is shared with the facilitator.
- Encrypt (M, FAP, GP, PK) → CT: The facilitator gets the public keys from the nodes in the AS. The facilitator then generates various FAPs based on the nodes in the AS, and communicates it to the client, along with the global parameters. The client selects a desirable FAP, encrypts the message with the FAP into a ciphertext. The ciphertext is then sent back to the facilitator.
- KeyGen(GP, GID, SK, t) → $K_{t,GID}$: When an answer has been found, the node in the AS that found it commences the key generation process. Each node in the AS must maintain its attribute and must be able to distribute a key reflecting that attribute to the facilitator. The keygen algorithm demands an identity GID, the global parameter, the attribute t belonging to the AS node, and the SK for that node. It produces the key $K_{t,GID}$ that reflects the Tier it belongs to and sends it to the facilitator.
- Decrypt (CT, GP, $K_{t,GID}$) → M: Once the facilitator gathers the attributes that match the FAP, it sends the global parameters, the ciphertext, and the keys corresponding to the attributes to the recipient. The recipient can now decrypt the message.

VI. Performance Evaluation

In this section, we present the results for the proposed mechanism. We test this by conducting an extensive simulation.

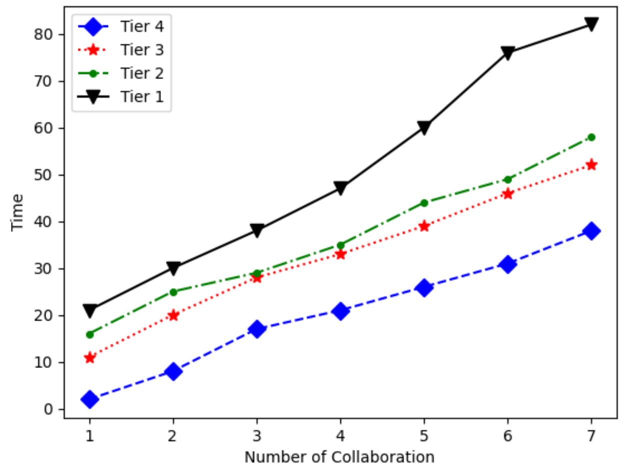


Figure 3. Tier Collaboration

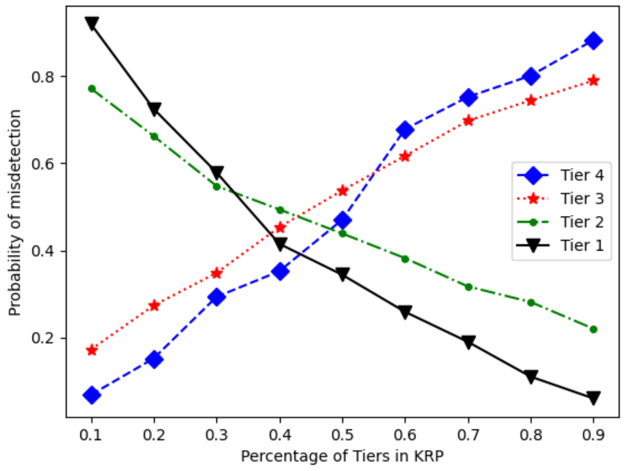


Figure 4. Misdetection

Our experiments were implemented in python and conducted on a workstation with intel 3.40 GHz CPU and 32 GB RAM running windows operating system. We simulate a network of 5 IGs (IoT, Stock, Sports, Food, Politics) where various nodes are accepted to the network based on the data they own. Each IG owns data that can be used to determine facts. As explained earlier, the nodes in the network have been ranked into Tiers based on the scoring process described in Section V. We ran experiments to detect false alarms, misdetections, and the time it takes for the collaboration between the nodes given an access policy. From Fig. 3, the experiment was to determine the time it takes for the collaboration between nodes in the same Tiers for the decryption of a message. We observe that nodes in Tier 1 take longer to collaborate with each other. The explanation for this is because Tier 1 level nodes are the least number of nodes in the network. As a consequence, this becomes a supply and demand issue, as the data of Tier 1 nodes are the most sought after. However, the ability of Tier 1 nodes to supply their services is hampered by their meager numbers. Fig. 4 aims to show the probability of misdetection by highlighting the outcomes of having various amounts of nodes belonging to

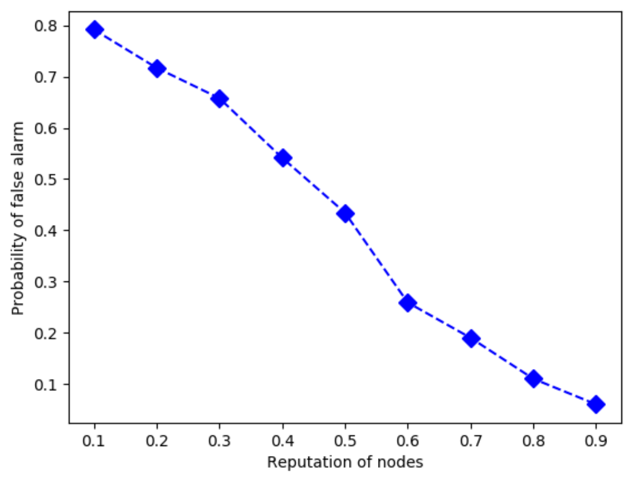


Figure 5. False Alarm

different Tiers present in the access policy. From the results, if the access policy contains nodes that are primarily from Tier 1, the chances of misdetection are significantly reduced. This represents a trade-off that shows that having more Tier 1 nodes in your FAP guarantees a higher confidence in the confirmation of an event, but this event confirmation process will not be confirmed right away or in real time. In Fig. 5, the false alarm reinforces the point of having higher Tier attributes in the FAP. From the plot, we determine that nodes with lower reputations may tend to assume their data can determine a fact. This is, however, usually not the situation as a FAP with only T4 attributes may not be able to confirm a fact. The plot shows that a higher amount of Tier 4 nodes in the answer set can result in the message never getting decrypted as these nodes might not have the full answer needed to decrypt the message even though they might have had the partial answers. This was a common theme when the threshold for the score employed in the acceptance of a node into the AS is lowered. A higher threshold connotes placing higher confidence in a node's ability to confirm an event accurately. Such trustworthy nodes are usually the nodes from Tier 1 and Tier 2. The prevalence of such nodes in the AS reduces the false alarm rate. As a result, an FAP that has primarily T1 or T2 nodes are more resilient to adversarial attacks (data falsification).

VII. Conclusion

In this work, we presented a variant of ABE, called EBE that aims to decrypt messages in the future when the occurrence of an event is confirmed. We illustrate how EBE can be employed by presenting a scenario where a will is decrypted when an individual has been confirmed dead. We discuss the decentralized data sharing approach on which EBE is implemented on. This decentralized data-sharing network is powered by the blockchain technology which ensures data undergoes a thorough vetting process before it is accepted to the network to avoid adversarial attacks. Specifically, we used the vetted data as a knowledge base in our work to confirm the occurrence of an event. We incorporated various NLP techniques and modified an existing ABE scheme to design this variant of ABE we call EBE.

Acknowledgments

This work is partly supported by the U.S. NSF under grants CNS 1650831, CCF-0939370 and HRD 1828811, and by the U.S. Department of Homeland Security under grant DHS 2019-ST-062-000003. However, any opinion, finding, and conclusions or recommendations expressed in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the funding agencies.

References

[1] G. Wood *et al.*, "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum project yellow paper*, vol. 151, no. 2014, pp. 1–32, 2014.

[2] D. Boneh, A. Sahai, and B. Waters, "Functional encryption: Definitions and challenges," in *Theory of Cryptography Conference*, pp. 253–273, Springer, 2011.

[3] A. Sahai and B. Waters, "Fuzzy identity-based encryption," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 457–473, Springer, 2005.

[4] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute-based encryption," in *2007 IEEE symposium on security and privacy (SP'07)*, pp. 321–334, IEEE, 2007.

[5] M. O. Rabin and C. Thorpe, "Time-lapse cryptography," 2006.

[6] I. F. Blake and A. C.-F. Chan, "Scalable, server-passive, user-anonymous timed release public key encryption from bilinear pairing.," *IACR Cryptology ePrint Archive*, vol. 2004, p. 211, 2004.

[7] R. L. Rivest, A. Shamir, and D. A. Wagner, "Time-lock puzzles and timed-release crypto," 1996.

[8] G. Zyskind, O. Nathan, and A. Pentland, "Enigma: Decentralized computation platform with guaranteed privacy," *arXiv preprint arXiv:1506.03471*, 2015.

[9] J. Ge, J. Ning, and A. Yu, "Cybervein: A dataflow blockchain platform," 2018.

[10] OpenMined, "Openmined: Building safer artificial intelligence.," 2018.

[11] R. Craib, R. Bradway, X. Dunn, and J. Krug, "Numeraire : A cryptographic token for coordinating machine intelligence and preventing overfitting," 2017.

[12] R. Gilad-Bachrach, N. Dowlin, K. Laine, K. Lauter, M. Naehrig, and J. Wernsing, "Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy," in *International Conference on Machine Learning*, pp. 201–210, 2016.

[13] H. Turesson, A. Roatis, H. Kim, and M. Laskowski, "Deep learning models as proof-of-useful work: A smarter, utilitarian scheme for achieving consensus on a blockchain," 2018.

[14] R. Doku, D. B. Rawat, and C. Liu, "Towards federated learning approach to determine data relevance in big

data," in *2019 IEEE 20th International Conference on Information Reuse and Integration for Data Science (IRI)*, pp. 184–192, 2019.

[15] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system. bitcoin," 2009.

[16] D. B. Rawat, V. Chaudhary, and R. Doku, "Blockchain: Emerging applications and use cases," *arXiv preprint arXiv:1904.12247*, 2019.

[17] Y. Sompolinsky and A. Zohar, "Accelerating bitcoin's transaction processing," *Fast Money Grows on Trees, Not Chains*, 2013.

[18] J. C. Corbett, J. Dean, M. Epstein, A. Fikes, C. Frost, J. J. Furman, S. Ghemawat, A. Gubarev, C. Heiser, P. Hochschild, *et al.*, "Spanner: Google's globally distributed database," *ACM Transactions on Computer Systems (TOCS)*, vol. 31, no. 3, p. 8, 2013.

[19] L. Luu, V. Narayanan, C. Zheng, K. Baweja, S. Gilbert, and P. Saxena, "A secure sharding protocol for open blockchains," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pp. 17–30, ACM, 2016.

[20] M. Zamani, M. Movahedi, and M. Raykova, "Rapidchain: A fast blockchain protocol via full sharding,"

[21] X. Li and D. Roth, "Learning question classifiers," in *Proceedings of the 19th international conference on Computational linguistics-Volume 1*, pp. 1–7, Association for Computational Linguistics, 2002.

[22] A. A. Selcuk, E. Uzun, and M. R. Pariente, "A reputation-based trust management system for p2p networks," in *IEEE International Symposium on Cluster Computing and the Grid, 2004. CCGrid 2004.*, pp. 251–258, IEEE, 2004.

[23] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-based encryption for fine-grained access control of encrypted data," in *Proceedings of the 13th ACM conference on Computer and communications security*, pp. 89–98, 2006.

[24] A. Lewko and B. Waters, "Decentralizing attribute-based encryption," in *Annual international conference on the theory and applications of cryptographic techniques*, pp. 568–588, Springer, 2011.

[25] M. Chase, "Multi-authority attribute based encryption," in *Theory of Cryptography Conference*, pp. 515–534, Springer, 2007.