A Survey on LIME: A Generic Data Lineage Framework

Sushma Nallamalli[1], Chandrajit Yadav Loya[2], Mohana Vamsi Priya Murala[3], Bhanu Prakash Doppala[4]

[1]Associate Professor, [2]Associate Professor & HOD, [3,4]Assistant Professor
Department Of computer Science and Engineering
DMS SVH College Of Engineering[1,2,3], Machilipatnam, Andhra Pradesh
Vignan's Institute of Information Technology[4], Visakhapatnam, Andhra Pradesh

***Abstract*: Data lineage is defined as a data life cycle that includes the data's origins and where it moves over time. It describes what happens to data as it goes through diverse processes and also provides visibility into the analytics pipeline which simplifies tracing errors, back to their sources. It also enables replaying specific portions or inputs of the dataflow for step-wise debugging or regenerating lost output. Data Leakage is the unapproved communication of data (or information) from source to an external destination. In this paper, we examine and submit a survey report on generic data lineage framework LIME for data flow across multiple entities that take principal roles (i.e., owner and consumer).The exact security guarantees required by such a data lineage mechanism toward identification of a guilty entity, by identifying and simplifying non-repudiation and honesty assumptions are presented. By using oblivious transfer, robust watermarking, and signature primitive's development and analyzing the data transfer protocol in a malicious environment between two entities is explained. Finally, we analyze an experimental evaluation to demonstrate the practicality of the protocol and apply the framework to the important data leakage scenarios of data outsourcing and social networks. In this work, we examine a watermarking based scheme to claim the identity of the leaker even if the data passes through multiple agents from data source to data usage centers.**

***Keywords*: Data Leakage, Data Provenance, Watermarking, Data Privacy, Security and Privacy, Obvious transfer, robust watermarking, Signature primitives and Accountability.**

1. Introduction

The value of the data is incredible, so it should not be leaked or altered. In the field of IT, huge database is being used. This database is shared with multiple people at a time. But during this sharing of the data, there is a chance of data vulnerability, leakage or alteration. Data leakage is defined as the accidental or unintentional distribution of private or sensitive data to unauthorized parties. Sensitive data of companies and organizations includes intellectual property (IP), financial information, patient information, personal credit-card data, and other information depending on the business and the industry like organization's internal process documents, strategic business plans, intellectual property, financial statements, security policies, network diagrams, blueprints etc.. Furthermore, in many cases, sensitive data is shared among various stakeholders such as employees working from outside the organizational premises (e.g., on laptops), business partners and customers. This increases the risk of confidential information falling into unauthorized hands. Whether caused by malicious intent, or an inadvertent mistake, by an insider or outsider, exposed sensitive information can seriously hurt an organization. The potential damage and adverse consequences of a data leak incident can be classified into the following two categories: direct and indirect loss. Direct loss refers to tangible damage that is easy to measure and estimate quantitatively. Indirect loss, on the other hand, is much harder to quantify and has a much broader impact in terms of cost, place and time [Bunker, 2009]. Direct loss includes violating regulations (such as those protecting customer privacy) resulting in fine/settlement/customer compensation fees; litigation of lawsuits; loss of future sales; costs of investigation and remedial/restoration fees. Indirect loss includes reduced share-price as a result of the negative publicity; damage to company's goodwill and reputation; customer abandonment; and exposure of Intellectual Property (business plans, code, financial reports, and meeting agendas) to competitors. In the course of doing business, sometimes sensitive data must be handovered to trusted third parties.

For example, a hospital may give patient records to researchers who will devise new treatments. Similarly, a company may have partnerships with other companies that require sharing customer data. Large amounts of digital data can be copied at almost no cost and can be spread through the internet in very short time. If the data is leaked, the data source centers like hospitals and health care centers must have a way to find the agent who leaked the data. The data can be either text or image data.

According to an interesting chronology of data breaches maintained by the Privacy Rights Clearinghouse (PRC), in the United States alone, 868;045;823 records have been breached from 4;355 data breaches made public since 2005 [2]. It is not hard to believe that this is just the tip of the iceberg, as most cases of information leakage go unreported due to fear of loss of customer confidence or regulatory penalties: it costs companies on average $214 per compromised record [3]. Large amounts of digital data can be copied at almost no cost and can be spread through the internet in very short time. Additionally, the risk of getting caught for data leakage is very low, as there are currently almost no accountability mechanisms. For these reasons, the problem of data leakage has reached a new dimension nowadays.

Not only companies are affected by data leakage, it is also a concern to individuals. The rise of social networks and smart phones has made the situation worse. In these environments, individuals disclose their personal information to various service

providers, commonly known as third party applications, in return for some possibly free services. With billions of mobile users worldwide, it is indeed a potentially huge market for advertising. These users spend significant time browsing the different multimedia and gaming capabilities of their devices, making them more exposed. Also, these devices now come with Wi-Fi and 3G, meaning they can be reached virtually everywhere. Add to this GPS capability and computing user preferences, and a new level of targeted advertising can be attained. Personalized ads that can match user's references with products and services in their interest have much higher chances of succeeding in capturing this user's attention and achieving better customer satisfaction, consequently increasing the profitability of ads. The same aspects that make these devices great platforms for advertising also impose leakage of private data like contacts information and calendar entries. In these environments, individuals disclose their personal information to various service providers, commonly known as third party applications, in return for some possibly free services. In the absence of proper regulations and accountability mechanisms, many of these applications share individuals' identifying information with dozens of advertising and Internet tracking companies. Hence, proper use and confidentiality of this data should be respected.

Even with access control mechanisms, where access to sensitive data is limited, a malicious authorized user can publish sensitive data as soon as he receives it. Primitives like encryption offer protection only as long as the information of interest is encrypted but once the recipient decrypts a message, nothing can prevent him from publishing the decrypted content. Thus it seems impossible to prevent data leakage proactively.

The data leakage detection industry is very heterogeneous as it of ripe product lines of leading IT security vendors. The technologies such as firewalls, encryption, access control, identity management, machine learning content based detectors and others have already been to offer protection against various factors of the data leakage threat.

Privacy, consumer rights, and advocacy organizations such as PRC [4] and EPIC [5] try to address the problem of information leakages through policies and awareness. How-ever, as seen in the following scenarios the effectiveness of policies is questionable as long as it is not possible to provably associate the guilty parties to the leakages.

Scenario 1:

Social networking: It was reported that third party applications of the widely used online social network (OSN) Facebook leak sensitive private information about the users or even their friends to advertising companies [6]. In this case, it was possible to determine that several applications were leaking data by analyzing their behavior and so these applications could be disabled by Facebook. However, it is not possible to make a particular application responsible for leakages that already happened, as many different applications had access to the private data.

Scenario 2:

Outsourcing: Up to 108;000 Florida state employees were informed that their personal information has been compromised due to improper outsourcing [7]. The outsourcing company that was handed sensitive data hired a further subcontractor that hired another subcontractor in India itself. Although the offshore subcontractor is suspected, it is not possible to provably associate one of the three companies to the leakage, as each of them had access to the data and could have possibly leaked it.

We find that the above and other data leakage scenarios can be associated to an absence of accountability mechanisms during data transfers. Leakers either do not focus on protection, or they intentionally expose confidential data without any concern, as they are convinced that the leaked data cannot be linked to them. In other words, when entities know that they can be held accountable for leakage of some information, they will demonstrate a better commitment towards its required protection.

In some cases, identification of the leaker is made possible by forensic techniques, but these are usually expensive and do not always generate the desired results. Therefore, we point out the need for a general accountability mechanism in data transfers. This accountability can be directly associated with provably detecting a transmission history of data across multiple entities starting from its origin. This is known as data provenance, data lineage or source tracing. The data provenance methodology, in the form of robust watermarking techniques [8] or adding fake data [9], has already been suggested in the literature and employed by some industries. However, most efforts have been ad-hoc in nature and there is no formal model available. Additionally, most of these approaches only allow identification of the leaker in a non-provable manner, which is not sufficient in many cases.

Contributions in LIME

In this paper, we examine this problem of provably associating the guilty party to the leakages, and work on the data lineage methodologies to solve the problem of information leakage in various leakage scenarios.

As first contribution, LIME is defined, a generic data lineage framework for data flow across multiple entities in the malicious environment. We observe that entities in data flows assume one of two roles: owner or consumer. An additional role in the form of auditor is introduced, whose task is to determine a guilty party for any data leak, and define the exact properties for communication between these roles. In the process, we an optional non-repudiation assumption is made between two owners and an optional trust (honesty) assumption is made by the auditor about the owners.

The key advantage of this model is that it enforces accountability by design; i.e., it drives the system designer to consider possible data leakages and the corresponding accountability constraints at the design stage. This helps to overcome the existing situation where most lineage mechanisms are applied only after a leakage has happened.

As second contribution, an accountable data transfer protocol is presented to verifiably transfer data between two entities. To deal with an untrusted sender and an untrusted receiver scenario associated with data transfer between two consumers, the protocols employ an interesting combination of the robust watermarking, oblivious transfer, and signature primitives.

The protocol is implemented as a C++ library by using the pairing-based cryptography (PBC) library [10] to build the underlying oblivious transfer and signature primitives. The image is chosen as a representative document type and use the Cox algorithm for robust image watermarking [11]. Water making is a mechanism to claim the identity of leaker. In this work, we employ the watermarking mechanism to claim the identify of leaked agent for text and image data. The analysis of storage, communication and computation overheads of the protocol, performance analysis and practicality demonstrations are given at the end. To simulate longer data transfer chains, experiments with multiple iterations of the implementation are performed and founded to be robust. Finally, the usage of the protocol to real-life data transfer scenarios is demonstrated such as online social networks and outsourcing.

2. Literature Survey

Panagiotis Papadimitriou and Hector Garcia- Molina[9] of Stanford University developed model for data leakage detection for accessing guiltiness of agents. The data distributor has to identify the malicious agent that leaked the information. In addition, they argue that current watermarking techniques are not practical, as they may embed extra information which could affect agents' work and their level of robustness may be inadequate. Existing matchmaker algorithm is unable to take correct decision based on QoS parameters.

Hasan, Sion and Winslett [13] - They summarize the data provenance as a system that enforces logging of read and write actions in a tamper-proof provenance chain. This creates the possibility of verifying the origin of information in a document. However, as an attacker is able to strip of the provenance information of a file, the problem of data leakage in malicious environments is not tackled by their approach.

A. Pretschner, M. Hilty, F. Sch¨utz, C. Schaefer, and T. Walter[14] employ data usage control enforcement systems and preventive measures to ensure that data is transferred in distributed systems in a controlled manner preserving the well defined policies. Handling global constraints, can lead to poor performance rendering inappropriate for applications with dynamic and real time requirements.

N.P. Sheppard, R. Safavi-Naini, and P. Ogunbona[15] find out the problem of an insider attack, where the data generator consists of multiple single entities and one of these publishes a version of the document. Usually methods for proof-of-ownership or fingerprinting are only applied after completion of the generating process, so all entities involved in the generation process have access to the original document and could possibly publish it without giving credit to the other authors, or also leak the document without being tracked.: The problem can be solved by the usage of watermarking and possibly even by using complete fingerprinting protocols during the generating phase of the document.

I. J. Cox, J. Kilian, F. T. Leighton, and T. Shamoon[11] in their paper 'Secure Spread Spectrum Watermarking for Multimedia' present a secure (tamper-resistant) algorithm for watermarking images, and a methodology for digital watermarking that may be generalized to audio, video, and multimedia data. We advocate that a watermark should be constructed as an independent and identically distributed (i.i.d.). In these cases, the watermark detector unambiguously identifies the owner. Further, the use of Gaussian noise, ensures strong resilience to multiple-document, or collusion, attacks.

B. Pfitzmann and M. Waidner[16] in the paper Asymmetric Fingerprinting for Larger Collusions explains that Fingerprinting schemes deter people from illegal copying of digital data by enabling the merchant of the data to identify the original buyer of a copy that was redistributed illegally. All known fingerprinting schemes are symmetric in the following sense: Both the buyer and the merchant know the fingerprinted copy. Thus, when the merchant finds this copy somewhere, there is no proof that it was the buyer who put it there, and not the merchant. They introduce asymmetric fingerprinting, where only the buyer knows the fingerprinted copy, and the merchant, upon finding it somewhere, can find out and prove to third parties whose copy it was.

S. Goldwasser, S. Micali, and R. L. Rivest[17] in the paper A Digital Signature Scheme Secure against Adaptive chosen-message attacks present a digital signature scheme based on the computational difficulty of integer factorization. The scheme possesses the novel property of being robust against an adaptive chosen-message attack: an adversary who receives signatures for messages of his choice (where each message may be chosen in a way that depends on the signatures of previously chosen messages) cannot later forge the signature of even a single additional message.

A. Adelsbach, S. Katzenbeisser, and A.-R. Sadeghi[18] in the paper A Computational Model for Watermark Robustness Multimedia security strategies generally combine cryptographic strategies with data hiding procedures like steganography or watermarking. Example appliances are dispute resolving, proof of ownership, (asymmetric/anonymous) fingerprinting and zero-knowledge watermark detection.

The need for formal security definitions of watermarking schemes is manifold, whereby the core need is to provide suitable abstractions to develop analyze and prove the security of applications on top of watermarking schemes. Although there exist formal models and definitions for information-theoretic and computational security of cryptographic and steganographic schemes, they cannot simply be adapted to watermarking schemes due to the fundamental differences among these approaches. Moreover, the existing formal definitions for watermark security still suffer from conceptual deficiencies.

Poh[19] addresses the problem of accountable data transfer with untrusted senders using the term fair content tracing. He presents a general framework to compare different approaches and splits protocols into four categories depending on their utilization of trusted third parties, i.e., no trusted third parties, offline trusted third parties, online trusted third parties and trusted hardware. Furthermore, he introduces the additional properties of recipient anonymity and fairness in association with payment.

3. Related Work

Data usage control enforcement systems employ preventive measures to ensure that data is transferred in distributed systems in a controlled manner preserving the well-defined policies. Techniques have been developed for securely distributing data by forming coalitions among the data owners. In controlled environments, such techniques can be composed with the protocols to improve data privacy.

In LIME[1], a model for accountable data transfer across multiple entities is proposed. Authors define participating parties, their interrelationships and give a concrete instantiation for a data transfer protocol using a novel combination of oblivious transfer, robust watermarking and digital signatures. But LIME works only for image data.

Controlled data disclosure is a well-studied problem in the security literature, where it is addressed using access control mechanisms. Although these mechanisms can control release of confidential information and also prevent accidental or malicious destruction of information, they do not cover propagation of information by a recipient that is supposed to keep the information private. For example, once an individual allows a third party app to access her information from a social network, she can no longer control how that app may redistribute the information. Therefore, the prevalent access control mechanisms are not adequate to resolve the problem of information leakages

LIME approach and watermarking are similar in the sense of providing agents with some kind of receiver identifying information. However, by its very nature, a watermark modifies the item being watermarked. Imperceptible to the human senses yet easily recognized by special software detectors, a digital watermark remains constant even through recording, manipulation and editing, compression and decompression, encryption, decryption and broadcast affecting the quality of the content. Digital watermarks are digital security features that transform multiple, previously passive elements of driver licenses, such as photo and artwork, into machine readable security tokens. When applied as a convert layer of security to driver licenses, digital watermarks enable fast, machine readable authentication of ID's. The features are imperceptible to humans, but read by computers or other devices enabled with special secure software.

If the object to be watermarked cannot be modified, then a watermark cannot be inserted. In such cases, methods that attach watermarks to the distributed data are not applicable. Finally, there are also lots of other works on mechanisms that allow only authorized users to access sensitive data through access control policies. Such approaches prevent in some sense data leakage by sharing information only with trusted parties. However, these policies are restrictive and may make it impossible to satisfy agent's requests.

LIME (Lineage In the Malicious Environment) can be used with any type of data for which watermarking schemes exist. Most watermarking schemes are designed for multimedia files such as images, videos, and audio files. In these multimedia files, watermarks are usually embedded by using a transformed representation (e.g. discrete cosine, wavelet or Fourier transform) and modifying transform domain coefficients. Watermarking techniques have also been developed for other data types such as relational databases, text files and even Android apps. The first two are especially interesting, as they allow to apply LIME to user databases or medical records. Watermarking relational databases can be done in different ways. The most common solutions are to embed information in noise-tolerant attributes of the entries or to create fake database entries. For watermarking of texts, there are two main approaches. The first one embeds information by changing the text's appearance (e.g. changing distance between words and lines) in a way that is imperceptible to humans. The second approach is also referred to as language watermarking and works on the semantic level of the text rather than on its appearance. A mechanism also has been proposed to insert watermarks to Android apps.

This mechanism encodes a watermark in a permutation graph and hides the graph as a linked list in the application. Due to the list representation, watermarks are encoded in the execution state of the application rather than in its syntax, which makes it robust against attacks. In this approach the authors propose to rather remove existing information than adding new information or modifying existing information. Thereby the watermarking scheme guarantees that no false entries are introduced. The above schemes can be employed in LIME framework to create data lineage for documents of the respective formats. The only modification that might be necessary when applying this scheme to a different document type is the splitting algorithm. For example for images it makes more sense to take small rectangles of the original image instead of simply taking the consecutive bytes from the pixel array. Embedding multiple watermarks into a single document has been discussed in literature and there are different techniques available. In that they discuss multiple re-watermarking and focus on segmented watermarking. It proves that multiple watermarking is possible which is very important for LIME scheme, as it allows to create a lineage over multiple levels. It would be desirable not to reveal the private watermarking key to the auditor during the auditor's investigation, so that it can be safely reused, but as discussed in current public key watermarking schemes are not secure and it is doubtful if it is possible to design one that is secure.

Sadeghi[18] presents approaches to zero-knowledge watermark detection. With this technology it is possible to convince another party of the presence of a watermark in a document without giving any information about the detection key or the watermark itself. However, the scheme discussed also hides the content of the watermark itself and are therefore unfit for LIME, as the auditor has to know the watermark to identify the guilty person. Furthermore, using a technology like this would come with additional constraints for the chosen watermarking scheme.

4. The LIME Framework

The *data leakage* is the general case that is addressed in data transfer settings, the simplified model LIME (Lineage in the malicious environment) is discussed. With LIME a clearly defined roles are assigned to each involved party and the inter-

relationships between these roles are defined in specific. This allows defining the exact properties that the transfer protocol has to fulfill in order to allow a provable identification of the guilty party in case of data leakage.

4.1 Model

As LIME is a general framework and should be applicable to all cases, we abstract the data type and call every data item *document*. There are three different roles that can be assigned to the involved parties in LIME: data *owner*, data *consumer* and *auditor*.

- The data owner is responsible for the management of documents.
- The consumer receives documents and can carry out some task using them.
- The auditor is not involved in the transfer of documents, he is only invoked when a leakage occurs and then performs all steps that are necessary to finding the leaker.

All of the mentioned roles can have multiple acceptations when the model is applied to a concrete setting. The reference to concrete acceptations of the model is called as *scenario*.

In typical scenarios the owner transfers documents to consumers. However, it is also possible that consumers pass on documents to other consumers or that owners exchange documents with each other. In the outsourcing scenario [7] the employees and their employer are owners, while the outsourcing companies are untrusted consumers.

Some trust assumptions are assumed for the implementation of the protocol. These trust assumptions are only used because they are realistic in a real world scenario and because it allows having a more efficient data transfer in the architecture.

When documents are transferred from one owner to another one, we can assume that the transfer is governed by a *non-cancellation assumption*. This means that the sending owner trusts the receiving owner to take responsibility if he should leak the document. As we consider consumers as mistrustful participants in the model, a transfer involving a consumer cannot be based on a non-cancellation assumption. Therefore, whenever a document is transferred to a consumer, the sender embeds information that uniquely findings the recipient. We call this *fingerprinting*. If the consumer leaks this document, it is possible to identify him with the help of the embedded information.

As presented, LIME relies on a technique for embedding identifiers into documents, as this provides an instrument to identify consumers that are responsible for data leakage. We require that the embedding does not affect the utility of the document. Furthermore, it should not be possible for a intentionally harmful consumer to remove the embedded information without rendering the document useless. A technique that can offer these properties is robust *watermarking*.

A key position in LIME is taken by the auditor. He is not involved in the transfer, but he takes action once a leakage occurs. He is invoked by an owner and provided with the leaked data. If the leaked data was transferred using this model, there is identifying information embedded for each consumer who received it.

Using this information the auditor can create an ordered chain of consumers who received the document. We call this chain the *descent* of the leaked document. The last consumer in the descent is the leaker. In the process of creating the descent each consumer can reveal new embedded information to the auditor to point to the next consumer and to prove his own innocence. .

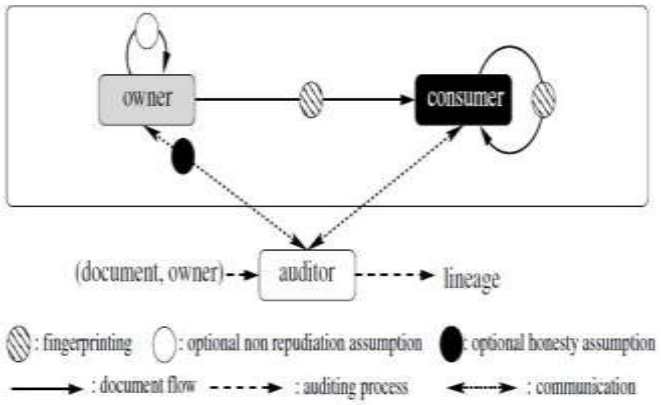

Fig. 1. The LIME framework: The framed box shows document transfers between owners and consumers. The auditor is a special entity which is only required when a descent occurs. The auditor then reconstructs the data descent by communicating with the involved parties.

In order to create a complete descent it is necessary that the auditor receives information from the owner, as only the owner can reveal the information embedded during the first transfer. We assume that the auditor is always invoked by the owner or that he is at least provided with information about the owner's identity, so that the auditor can start his investigation with the owner and a complete descent can be created.

We can assume that the auditor trusts the owner to be honest. Honesty in this case means that the owner does not leak a document and blame another party. We can make this assumption as the owner is concerned with the document's privacy in the first place. However, the auditor does not trust the consumers. In a real world setting the auditor can be any authority, for example

a governmental institution, police, a legal person or even some software. In the outsourcing scenario [7], the employer can invoke the auditor who recreates the descent and thereby uncovers the finding of the leaker

The employer can use this information to take legal actions against the outsourcing company. We show the flow of documents, the optional non-repudiation and honesty assumptions as well as the cases in which fingerprinting is used in Fig. 1.

These honesty and non-repudiation assumptions are chosen because they realistically model many real-world scenarios. Due to these assumptions we can reduce the overhead introduced to transfers by LIME: In a transfer between two owners no fingerprinting has to be used and the owner can run a simplified transfer protocol because he is trusted by the auditor.If an owner is untrusted we can easily treat him as a consumer, so we do not have to make any trust assumptions about him.

This work also motivates further research on data leakage detection techniques for various document types and scenarios. For example, it will be an interesting future research direction to design a verifiable lineage protocol for derived data.

We can apply LIME to user database or medical records as watermarking also supports data types like relational database, android applications and text files. We can create encrypted watermark to protect the data from leakage. For LIME watermarking is used on text by inserting or embedding information and changing the texts appearance, by changing the distance between words and lines or insert text into the image so that it can be invisible or inaudible to humans. Language watermarking scheme can also be used which does not work on appearance, it works on semantic level of data. A simple example of watermarked image is shown in Fig.2

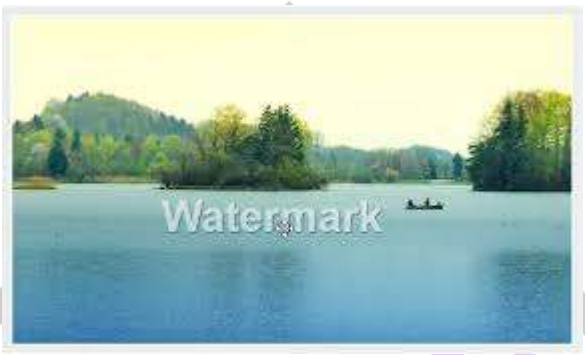

Fig.2 Example of Watermarked image

Watermarking techniques does not look in syntax when they are encoded in execution state of the application which makes the system robust. The usage of watermark is shown in Fig.3

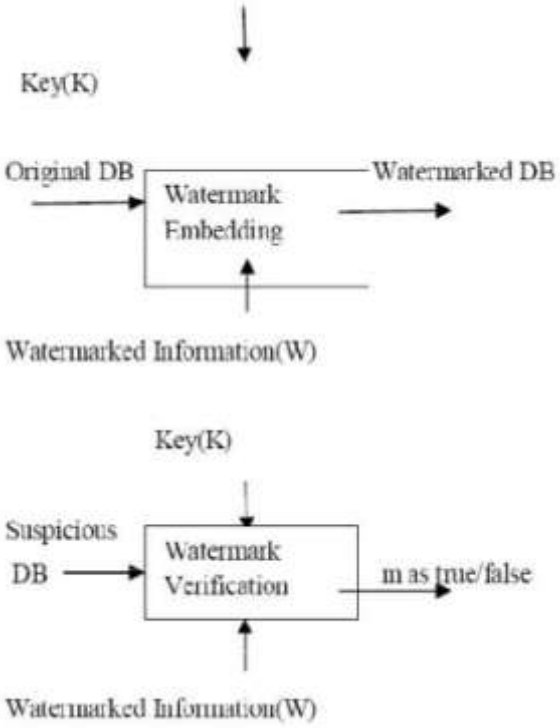

Fig.3 Processing of Watermarking

4.2 Threat Model and Design Goals

Although the problem of data leakage is addressed, LIME cannot guarantee that data leakage does not occur in the first place; once a consumer has received a document, nothing can prevent him from publishing it.

A method is designed to provably identify the guilty party once a leakage has been detected. By introducing this reactive accountability, we expect that leakage is going to occur less often, since the identification of the guilty party will in most cases lead to negative consequences.

Attackers in this model are the consumers that take every possible step to publish a document without being held accountable for their actions. As the owner does not trust the consumer, he uses fingerprinting every time he passes a document to a consumer. However, we assume that the consumer tries to remove this identifying information in order to be able to publish the document safely. As already mentioned, consumers might transfer a document to another consumer, so we also have to consider the case of an untrusted sender. A different problem that arises with the possibility of false accusation is denial. If false accusation is possible, then every guilty receiving consumer can claim that he is innocent and was framed by the sending consumer.

The crucial phase in the model is the transfer of a document involving untrusted entities, so we clearly define which properties we require the protocol to fulfill. We call the two parties sender and recipient. We expect a transfer protocol to fulfill the following properties and only tolerate failures with negligible probabilities.

- *Correctness:* When both parties follow the protocol steps correctly and only publish their version of the document, the guilty party can be found.
- *No framing:* The sender cannot frame recipients for the sender's leakages.
- *No denial:* If the recipient leaks a document, he can be provably associated with it.

We also require the model to be collusion resistant, i.e., it should be able to tolerate a small number of colluding attackers. We also assume that the communication links between parties are reliable.

Non-goals. This model does not aim at proactively stopping data leakage, it only provide means to provably identify the guilty party in case a leak should occur, so that further steps can be taken. This model also does not aim for integrity, as at any point an entity can decide to exchange the document to be sent with another one. This approach does not account for derived data (derived data can for example be generated by applying aggregate functions or other statistical operations), as much of the original information can be lost during the creation process of derived data.

4.3 Primitives

In this scheme we make use of digital signatures. More precisely, we use a CMA-secure signature [12], i.e., no polynomial-time adversary is able to forge a signature with non-negligible probability. For a message m that has been signed with party A's signing key, we write ½m&$_{skA}$: We use a symmetric encryption scheme that offers security under chosen plaintext attacks, writing c ¼ encðm; ekÞ for encryption of a message m and m ¼ decðc; ekÞ for decryption of a ciphertext c with symmetric key ek.

4.3.1Robust Watermarking

For watermarking, we need a so-called similarity function *sim(D,D^1)* that returns ⊤ and if the two documents *D* and *D^1* are considered similar in the used context and ⊥ otherwise. For example, two images could be considered similar, if a human observer can extract the same information from them.

Let D be the set of all possible documents, *WM* the set of all possible watermarks, *K* the set of keys and *k* the security parameter of the watermarking scheme. A symmetric, detecting watermarking scheme is defined by three polynomial-time algorithms: *probabilistic Key Generation Algorithm, Embedding Algorithm, and Detection Algorithm.* We require the following properties: *Imperceptibility, Effectiveness, and Robustness.*

Additionally, we require the watermarking scheme to support multiple re-watermarking, i.e., it should allow for multiple (bounded by the dataflow path length) watermarks to be embedded successively without influencing their individual detectability. This property can also be considered as a special kind of robustness, as it prevents adversaries from making a watermark undetectable simply by adding more watermarks using the same algorithm. We also expect the watermarking scheme to be collusion resistant.

4.3.2 1-Out-of-2 Oblivious Transfer

1-out-of-2 Oblivious Transfer involves two parties, the sender and the chooser. The sender offers two items M_0 and M_1 and the chooser chooses a bit s. The chooser obtains M_s, but no information about M_{1-s} and the sender learns nothing regarding s. In this context, when speaking of learning nothing, we actually mean nothing can be learned with non-negligible probability.

When we use OT_1^2 in the protocols to send messages, the sender actually encrypts the messages, sends both cipher-texts to the chooser and performs OT_1^2 just on the decryption keys. This allows to use the OT_1^2 protocol with a fixed message size while actually sending messages of arbitrary size. Note that this could only be a security risk if the chooser was able to break the encryption scheme.

4.4 Accountable Data Transfer

In this section we specify how one party transfers a document to another one, what information is embedded and which steps the auditor performs to find the guilty party in case of data leakage. We assume a public key infrastructure to be present, i.e., both parties know each other's signature verification key.

4.4.1 Trusted Sender

In the case of a trusted sender it is sufficient for the sender to embed identifying information, so that the guilty party can be found. As the sender is trusted, there is no need for further security mechanisms. In Fig. 4, we present a transfer protocol that fulfills the properties of correctness and no denial. As the sender is trusted to be honest, we do not need the no framing property.

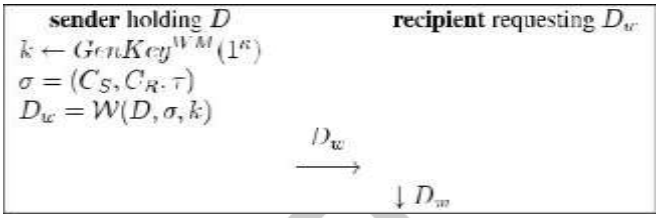

Fig.4. Protocol for trusted senders: The sender watermarks the original document with a signed statement containing the participants' identifiers and a timestamp, and sends the watermarked document to the recipient.

The sender, who is in possession of some document D, creates a watermarking key k, embeds a triple σ consisting of the two parties' identifiers and a time-stamp τ into D to create D_w. He then sends D_w to the recipient, who will be held accountable for this version of the document. As the sender also knows D_w, this very simple protocol is only applicable if the sender is completely trusted; otherwise the sender could publish D_w and blame the recipient.

4.4.2 Untrusted Sender

In the case of an untrusted sender we have to take additional actions to prevent the sender from cheating, i.e., we have to fulfill the no framing property. To achieve this property, the sender divides the original document into n parts and for each part he creates two differently watermarked versions. He then transfers one of each of these two versions to the recipient via OT_1^2. The recipient is held accountable only for the document with the parts that he received, but the sender does not know which versions they are. The probability for the sender to cheat is $1/2^n$.

First, the sender generates two watermarking keys k_1 and k_2. It is in his own interest that these keys are *fresh* and *distinct*. The identifying information that the sender embeds into the document D is a signed statement containing the sender's and recipient's identifiers and a timestamp τ, so that every valid watermark is authorized by the recipient. The sender computes the watermarked document D^1, splits the document D^1 into n parts and creates two different versions $D_{i,j}$ of each part by adding an additional watermark $j \epsilon \{0,1\}$. For each version of each part $D_{i,j}$ he creates a signed message $m_{i,j}$ containing the timestamp of the current transfer, the part's index and the content of the version's second watermark. Then he generates an AES key $ek_{i,j}$, encrypts $c_{i,j}$ and sends $c_{i,j}$ to the recipient. The recipient chooses a bit $b_i \epsilon \{0,1\}$ for each $i \epsilon \{1 \ldots n\}$and receives ek ,via oblivious transfer. He then decrypts and reconstructs the document by joining the parts $D_{1,b1} \ldots D_{n,bn}$. The signed statements serve as proof of his choice. As for each part he chooses a bit b_i, the final version is identified by a bitstring $b^1 \epsilon \{0,1\}^n$. As he will only be held accountable for the version watermarked with b^1, we have a failure probability (i.e., the sender correctly guessing b) of $1/2^n$.

4.4.3Data Lineage Generation

The auditor is the entity that is used to find the guilty party in case of a leakage. He is invoked by the owner of the document and is provided with the leaked document. In order to find the guilty party, the auditor proceeds in the following way:

[1] The auditor initially takes the owner as the current suspect.

[2] The auditor appends the current suspect to the lineage.

[3] The auditor sends the leaked document to the current suspect and asks him to provide the detection keys k_1 and k_2 for the watermarks in this document as well as the watermark s. If a non-blind water-marking scheme is used, the auditor additionally requests the unmarked version of the document.

[4] If, with key k_1, σ cannot be detected, the auditor continues with 9.

[5] If the current suspect is trusted, the auditor checks that σ is of the form (C_S,C_R,τ) where C_S is the identifier of the current suspect, takes C_R as current suspect and continues with 2.

[6] The auditor verifies that σ is of the form (C_S,C_R,τ) ,where C_S is the identifier of the current suspect. He also verifies the validity of the signature.

[7] The auditor splits the document into n parts and for each part he tries to detect 0 and 1 with key k_2. If none of these or both of these are detectable, he continues with 9. Otherwise he sets b^1_i as the detected bit for the ith part.

8) The auditor asks C_R to prove his choice of $b^1 = b_1 \ldots . b_n$ for the given timestamp τ . If C_R is not able to give a correct proof the auditor takes C_R as current suspect and continues with 2.

9) The auditor outputs the lineage. The last entry is responsible for the leakage.

Interestingly, AES performs all its computations on bytes rather than bits. Hence, AES treats the 128 bits of a plaintext block as 16 bytes. These 16 bytes are arranged in four columns and four rows for processing as a matrix. Unlike DES, the number of rounds in AES is variable and depends on the length of the key. AES uses 10 rounds for 128-bit keys, 12 rounds for 192-bit keys and 14 rounds for 256-bit keys. Each of these rounds uses a different 128-bit round key, which is calculated from the original AES key.

5. Implementation and Microbenchmarking

The protocol is implemented as a proof-of-concept and to analyze its performance.Oblivious transfer sub protocol is implemented by Naor and Pinkas (authors) using the PBC library, which itself makes use of the GMP library. For signatures the BLS scheme is implemented, also using the PBC library. For symmetric encryption AES is implemented from the Crypto++ library. For watermarking an implementation of the Cox algorithm for robust image watermarking from Peter Meerwald's watermarking toolbox is used. The α-factor, which determines the strength of the watermark, is set to a value of 0.1.

The experiment is executed with different parameters to analyze the performance. The sender and recipient part of the protocol are both executed in the same program, i.e., network sending is not analyzed, but only computational performance. The executing machine is a Lenovo ThinkPad model T430 with 8 GB RAM and 4 × 2.6GHz cores, but all executions were performed sequentially. The execution times are measured for different phases of the protocol: watermarking, signature creation, encryption, oblivious transfer and detection. The experiment is executed 250 times and the average computation time and the standard deviation are determined.

In the first experiment an image of size 512 × 512 pixels are used and the number of parts the image was split into. The results are shown in Fig. 5(a). We can see that the execution time of watermarking, signatures, oblivious transfer and detection is linear in the number of document parts. The execution time of encryption is also increasing slowly, but it is still insignificant compared to the other phases.

In the second experiment a fixed number of document parts of 256 re used and changed the size of the image used.The results can be seen in Fig. 5(b). We can observe that the execution time for watermarking and detection is linear in the image size, but in contrast to the previous result, we already start with a long execution time for small sizes and the growth is rather slow. The execution time for oblivious transfer stays constant, which might be surprising at first sight, but this can be explained easily, as we only transfer AES keys in the oblivious transfer phase and these are of constant size for all image sizes. Of course the encrypted files also have to be sent over the network (so there would be a higher communication overhead for bigger images), but in this experiment we only considered the computational costs. The execution time for the creation of the signatures is also constant as the number and form of the signed statements is the same for all images. Again the execution time of encryption is increasing slowly, but it is of no significance compared to the other phases.

Although we use only image files as documents in the experimental implementation, we stress that the same mechanism can be used for all types of data for which robust watermarking schemes exist.

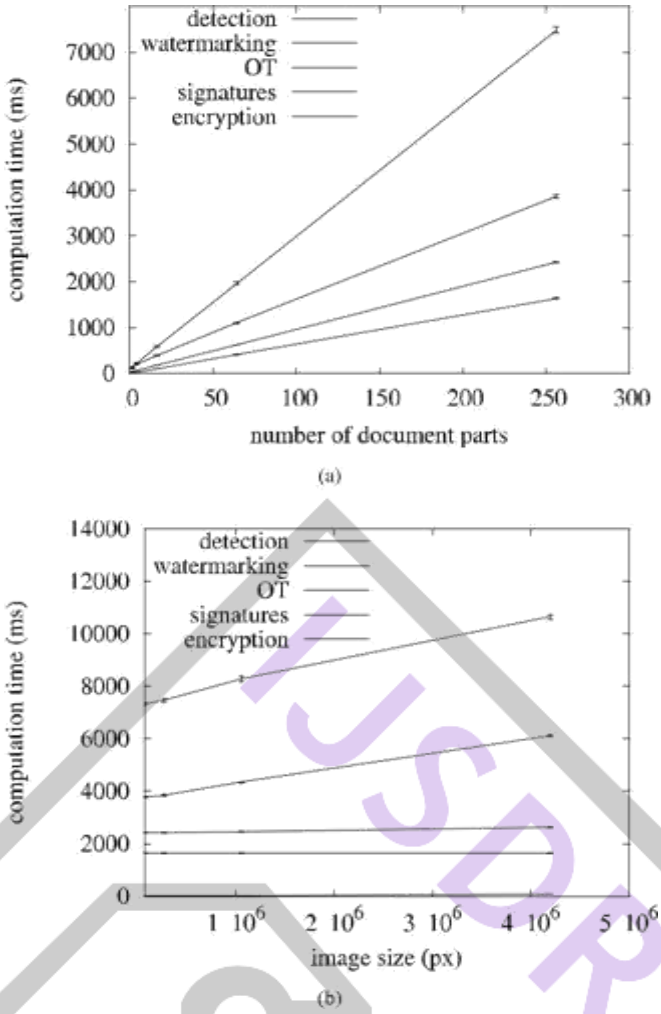

Fig.5.(a) shows computation times for different numbers of document parts;(b) shows computation times for different image sizes.

5.1 Communication Overhead

Assuming 128-bit security level for encryptions and signatures, and splitting the document of size *s* into *n* parts, we can compute the communication overhead as follows: First the recipient sends σ consisting of two identifiers (8 bytes), one Unix timestamp (8 bytes) and one BLS signature (32 bytes). The sender in return sends 2n times a document part of size *s/n* and a message consisting of one timestamp (8 bytes), one single bit, one integer (4 bytes) together with the according BLS signature (32 bytes). This totals to $2 \cdot (s + n \cdot (8 + 1 + 4 + 32)) = 2 \cdot (s + 45n)$ *bytes*. Additionally, sender and recipient run *n* parallel instances of an oblivious transfer protocol. In each protocol run, the sender sends two group elements (64 bytes) in the initialization phase. In the transfer phase the chooser sends one group element (32 bytes) and the sender sends two encryptions of messages (which are AES keys) (64 bytes). In total the sender sends $2s + 218n$ bytes and the recipient sends $32n + 48$ bytes. For example with an image of size s = 1MB and n = 64 a communication overhead of 2.008MB for the sender and 2.05KB for the recipient are found in practical.

5.2 Storage Overhead

Both parties need to store some data so that they can provide the necessary information to the auditor during the process of lineage generation. The sender needs to store the first watermark σ (48 bytes) and 2 watermarking keys. For a non-blind watermarking scheme like the Cox algorithm used in the implementation the sender also needs to store the original document.

6 . Scenarios

6.1 Outsourcing

The first diagram in Fig. 6 shows a typical outsourcing scenario. An organization acts as owner and can outsource tasks to outsourcing companies which act as consumers in LIME model. It is possible that the outsourcing companies receive sensitive data to work on and as the outsourcing companies are not necessarily trusted by the organization, fingerprinting is used on

transferred documents. The out-sourcing company itself can outsource tasks to other out-sourcing companies and thus relay the documents, again using fingerprinting. It is important to notice that a single organization can outsource to many different outsourcing companies in parallel, thus creating a tree-shaped transfer diagram. If now at any point one of the involved outsourcing companies leaks a confidential document, the organization can invoke the auditor to find the responsible party. The auditor then asks the organization to reveal the first set of fingerprints in the leaked document, which leads the auditor to one of the outsourcing companies. This out-sourcing company can in turn reveal additional finger-prints in the leaked document in order to point to the next outsourcing company and to prove its own innocence. Finally, the auditor creates the complete lineage and is able to determine the guilty party. The responsible party can be clearly found using LIME.

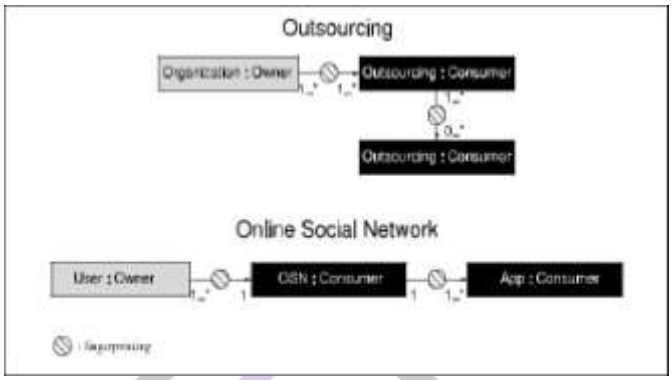

Fig. 6. Outsourcing scenario.

6.2 Online Social Network

The second diagram in Fig. 6 shows an online social net-working scenario. The users of the network are the owners, as they enter their personal information, post messages, etc. The online social network uses all this information as a consumer in this scenario. Third party applications that have access to this information in return for some service act as further consumers in this scenario. The users give their information to the OSN which can relay that information to third party applications using fingerprinting. In case of a leakage the auditor can create the lineage of the leaked document and thereby provably determine the responsible party.
In the introduction we gave an example where third party applications leaked private information of Face book users to advertising companies [5]. Although, using forensic methods, it was possible to determine which applications leaked data, given some leaked information it is not possible to find the responsible application. Using LIME this link can be found and proven.

7. Conclusion

We present the survey report on LIME a model for accountable data transfer across multiple entities. The participating parties, their interrelationships are clearly defined and a concrete instantiation for a data transfer protocol using a novel combination of oblivious transfer, robust watermarking and digital signatures are presented. The LIME correctness is proved by realizing and giving micro benchmarking results. By presenting a general applicable framework, accountability is introduced as early as in the design phase of a data transfer infrastructure. Although LIME does not actively prevent data leakage, it introduces reactive accountability. Thus, it will deter malicious parties from leaking private documents and will encourage honest (but careless) parties to provide the required protection for sensitive data. This system is flexible as difference between trusted senders (usually owners) and untrusted senders (usually consumers) is clearly shown. In the case of the trusted sender, a very simple protocol with little overhead is possible. The untrusted sender requires a more complicated protocol, but the results are not based on trust assumptions and therefore they should be able to convince a neutral entity. This work motivates further research on data leakage detection techniques for various document types and scenarios.

REFERENCES

[1] Michael Backes, Niklas Grimm, and Aniket Kate," Data Lineage in Malicious Environments" DOI 10.1109/TDSC.2015.2399296, IEEE. M.Backes, N.Grimm, and A.Kate, "Lime:Data lineage in the malicious environment," in Security and Trust Management-10th Interntional Workshop, STM2014, Wroclaw, Poland, September10- 11, 2014, Proceedings, 2014, pp.183-187.

[2] Transactions on Dependable and Secure ComputingChronology of data breaches [Online]. Available: http://www. privacyrights.org/data-breach, 2014.

[3] Data breach cost [Online]. Available: http://www. symantec. com/ about/ news/ release/ article.jsp?prid=20110308_01, 2011.

[4]Privacy rights clearinghouse [Online]. Available: http://www. privacyrights.org, 2014.
[5] (1994). Electronic privacy information center (EPIC) [Online]. Available: http://epic.org, 1994.

[6] Facebook in privacy breach [Online]. Available: http://online.wsj. com/article/SB10001424052702304772804575558484075236968. html, 2010.

[7] Offshore outsourcing [Online]. Available: http://www.computer-world.com/s/article/109938/Offshore_outsourcing_cited_in_Florida_data_leak, 2006.

[8] A. Mascher-Kampfer, H. Stogner,€ and A. Uhl, "Multiple re-water-marking scenarios," in Proc. 13th Int. Conf. Syst., Signals, Image Process., 2006, pp. 53–56.

[9] P. Papadimitriou and H. Garcia-Molina, "Data leakage detection," IEEE Trans. Knowl. Data Eng., vol. 23, no. 1, pp. 51–63, Jan. 2011.

[10] Pairing-based cryptography library (PBC) [Online]. Available: http://crypto.stanford.edu/pbc, 2014.

[11] I. J. Cox, J. Kilian, F. T. Leighton, and T. Shamoon, "Secure spread spectrum watermarking for multimedia," IEEE Trans. Image Process., vol. 6, no. 12, pp. 1673–1687, Dec. 1997.

[12] S. Goldwasser, S. Micali, and R. L. Rivest, "A digital signature scheme secure against adaptive chosen-message attacks," SIAM J. Comput., vol. 17, no. 2, pp. 281–308, 1988.

[13] R Hasan, R Sion, M Winslett ," Introducing secure provenance: problems and challenges"
Proceedings of the 2007 ACM workshop on Storage security and survivability

[14] Pretschner, M. Hilty, F. Schutz, C. Schaefer, and T. Walter, "Usage Control Enforcement: Present and Future,"IEEE Security \& Privacy, vol. 6, no. 4, pp. 44-53, 2008.

[15] NP Sheppard, R Safavi-Naini, P Ogunbona "On multiple watermarking"
Proceedings of the 2001 workshop on Multimedia and Security: new challenges, at ACM Multimedia, pages 3-6,Ottawa,Canada

[16] B Pfitzmann, M Waidner "Asymmetric fingerprinting for larger collusions" Proceedings of the 4th ACM conference on Computer and communications security

[17] S. Goldwasser, S. Micali, R.L. Rivest. *A Digital Signature Scheme Secure Against Adaptive Chosen-Message Attacks*. Siam J. Computing 17(2): 281-308 (1988).

[18] A. Adelsbach, S. Katzenbeisser, and A.-R. Sadeghi "A computational model for watermark robustness" Proceeding IH'06 Proceedings of the 8th international conference on information hiding pages 145-160,July 10-12,2006

[19] G. S. Poh. Design and Analysis of Fair Content Tracing Protocols. PhD thesis, 2009.