International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Index Copernicus Value (2013): 6.14 | Impact Factor (2015): 6.391

Data Hopping in Malicious Environment

S Mohammed Ghouse[1], Sheethal R[2]

[1]Assistant Professor, Department of Computer Science and Engineering, SEACET, Bangalore, India

[2]M. Tech, Computer Science and Engineering, SEACET, Bangalore, India

Abstract: ***Deliberate or inadvertent spillage of private information is without a doubt a standout amongst the most serious security dangers that associations face in the computerized period. The risk now stretches out to our own lives: a plenty of individual data is accessible to informal organizations and Smartphone suppliers and is in a roundabout way exchanged to deceitful outsider and fourth gathering applications. In this work, we show a non specific information ancestry structure LIME for information stream over various elements that take two trademark, important parts (i.e., proprietor and buyer). We characterize the definite security ensures required by such an information ancestry instrument toward distinguishing proof of a blameworthy element, and recognize the improving non-disavowal and genuineness suppositions. We then create and examine a novel responsible information exchange convention between two elements inside a pernicious domain by expanding upon unmindful exchange, strong watermarking, and mark primitives. At long last, we perform an exploratory assessment to show the reasonableness of our convention and apply our system to the critical information spillage situations of information outsourcing and interpersonal organizations. As a rule, we consider LIME, our ancestry system for information exchange, to be a key step towards accomplishing responsibility by outline***

Keywords: Smartphone, security threats, data lineage, LIME, robust watermarking, signature primitives and accountability

1. Introduction

In the digital era, information leakage through unintentional exposures, or intentional sabotage by disgruntled employees and malicious external entities, present one of the most serious threats to organizations. 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. 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. 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.

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. Privacy, consumer rights, and advocacy organizations such as PRC and EPIC try to address the problem of information leakages through policies and awareness. However, 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 Face book leak sensitive private information about the users or even their friends to advertising companies. 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 Face book. 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. 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.

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

2. Our Contributions

In this paper, we formalize 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 our first contribution, we define LIME, 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. We introduce an additional role in the form of auditor, 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 identify an optional non-repudiation assumption made between two owners, and an optional trust (honesty) assumption made by the auditor about the owners.

As our second contribution, we present an accountable data transfer protocol 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; our protocols employ an interesting combination of the robust watermarking, oblivious transfer, and signature primitives.

3. The Lime Framework

As we want to address a general case of data leakage in data trans-fer settings, we propose the simplifying model LIME (Lineage in the malicious environment). With LIME we assign a clearly de-fined role to each involved party and define the inter-relationships between these roles. This allows us to define the exact properties that our transfer protocol has to fulfill in order to allow a provable identification of the guilty party in case of data leakage.

Model

As LIME is a general model 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 and 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 identify the leaker. All of the mentioned roles can have multiple instantiations when our model is applied to a concrete setting. We refer to a concrete instantiation of our model 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 the employees and their employer are owners, while the outsourcing companies are untrusted consumers.

In the following we show relations between the different entities and introduce optional trust assumptions. We only use these trust assumptions because we find that they are realistic in a real world scenario and because it allows us to have a more efficient data transfer in our framework. At the end of this section we explain how our framework can be applied without any trust assumptions.

When documents are transferred from one owner to another one, we can assume that the transfer is governed by a *non-repudiation 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 untrusted participants in our model, a transfer involving a consumer cannot be based on a non-repudiation assumption. Therefore, whenever a document is transferred to a consumer, the sender embeds information that uniquely identifies 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 malicious 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 our model, there is identifying information embedded for each consumer who received it.

3

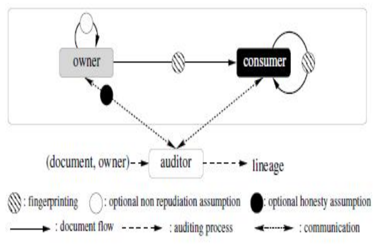

Fig. 1. The LIME framework: The framed box shows document transfers between owners and consumers. The auditor is a special entity which is only required when a leakage occurs. The auditor then reconstructs the data lineage by communicating with the involved parties.

Threat Model and Design Goals

Although we try to address the problem of data leakage, 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. We offer a method 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. As our only goal is to identify guilty parties, the attacks we are concerned about are those that disable the auditor from provably identifying the guilty party.

Therefore, we consider attackers in our model as 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 previously, consumers might transfer a document to another consumer, so we also have to consider the case of an *untrusted sender*. This is problematic because a sending consumer who embeds an identifier and sends the marked version to the receiving consumer could keep a copy of this version, publish it and so frame the receiving consumer. Another possibility to frame other consumers is to use fingerprinting on a document without even performing a transfer and publish the resulting document.

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 our model is the transfer of a document involving untrusted entities, so we clearly define which properties we require our 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.

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

We also require our 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. We do not aim at proactively stopping data leakage; we only provide means to provably identify the guilty party in case a leak should occur, so that further steps can be taken. We also do not aim for integrity, as at any point an entity can decide to exchange the document to be sent with another one. However, in our settings, the sender wants the receiver to have the correct document, as he expects the recipient to perform a task using the document so that he eventually obtains a meaningful result.

4. Primitives

A function f is *negligible* if for all c > 0 there is a n_c so that for all $n \geq n_c$ $f(n) \leq \frac{1}{n^C}$. In our 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 $[m]_{skA}$. We use a symmetric encryption scheme that offers security under chosen plaintext attacks, writing c = enc(m, ek) for encryption of a message m and m = dec(c, ek) for decryption of a cipher text c with symmetric key ek.

Robust Watermarking

We use the definition of watermarking by Adelsbach et al. To argue about watermarking, we need a so-called similarity function sim(D, D') that returns ⊤ if the two documents D and D' are considered similar in the used context and ⊥ otherwise. The similarity function is a different one for each data type used and we assume it is given. For example, two images could be considerered similar, if a human observer can extract the same information from them.

Let **D** be the set of all possible documents, **WM** ⊆ $\{0, 1\}^+$ the set of all possible watermarks, **K** the set of keys and κ the security parameter of the watermarking scheme. A *symmetric, detecting watermarking scheme* is defined by three polynomial-time algorithms:

Probabilistic *Key Generation Algorithm:*

$GenKey^{W\ M}(1^\kappa)$ outputs a key k ∈ **K** for a given security parameter κ.

- The probabilistic *Embedding Algorithm* generates a watermarked document D' = **W**(D, w, k) on input of the original document D ∈ **D**, the watermark w ∈ **WM** and the key k ∈ **K**.
- The *Detection Algorithm* Detect(D', w, D, k) outputs ⊤ or

⊥ on input of a (potentially watermarked) document D' ∈ **D**, a watermark w ∈ **WM**, the original document D ∈ **D** and a key k ∈ **K**. ⊤ means that the watermark is detectable; ⊥ means, that it is not.

We require the following properties:

- **Imperceptibility:** ∀D ∈ **D**, ∀w ∈ **WM**, ∀k ∈ **K**.D' ← **W**(D, w, k) ⇒ sim(D, D') = ⊤, i.e., the original document and the watermarked document are similar.
- **Effectiveness:** ∀D ∈ **D**, ∀w ∈ **WM**, ∀k ∈ **K**.D' ← **W**(D, w, k) ⇒ Detect(D', w, D, k) = ⊤, i.e., if a watermark is embedded using a key k, the same watermark should be detectable using the same key k.

Detect (D'', w, D, k) = ⊥ with non-negligible probability. This means that no adversary can efficiently remove or change a watermark without rendering the document unusable (i.e., breaking the similarity).

Additionally, we require our 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 delectability. 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. More information and some experimental results about this property can be found in.

It is shown that the scheme is robust against many common attacks such as scaling, JPEG compression, printing, Xeroxing and scanning, multiple re-watermarking and others.

1-out-of-2 Oblivious Transfer

1-out-of-2 Oblivious Transfer (OT_1^2) involves two parties, the *sender* and the *chooser*. The sender offers two items M_0 and M_1 and the chooser chooses a bit σ. The chooser obtains M_σ but no information about $M_{1-\sigma}$ and the sender learns anything regarding σ. 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 our 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 us to use the OT_1^2 protocol with a fixed message size

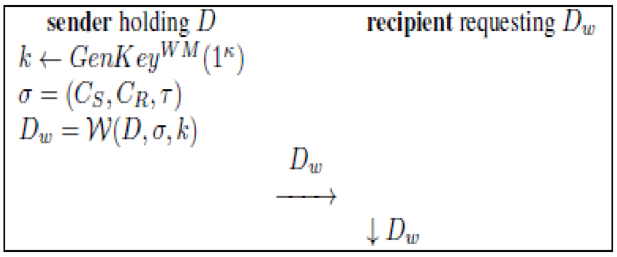
sender holding D
recipient requesting D_w
$k \leftarrow GenKey^{WM}(1^\kappa)$
$\sigma = (C_S, C_R, \tau)$
$D_w = W(D, \sigma, k)$
$\xrightarrow{D_w}$
$\downarrow D_w$

Fig. 2. protocol for trusted senders: The sender watermarks the original document with a signed statment containing the participants' identifiers and a timestamp, and sends the watermarked document to the recipient.

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. There are different concrete instantiations of this primitive. As an example implementation of $OT_1{}^2$ we show a protocol by Naor and Pinkas in the Appendix.

5. 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.

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. 2, we present a transfer protocol that fulfills the properties of correctness and no denial as defined in Section 2.2. As the sender is trusted to be honest, we do not need the any framing property.

The sender, who is in possession of some document D, creates a watermarking key k, embeds a triple σ = (C_S , C_R, τ) consisting of the two parties' identifiers and a timestamp τ into D to create D_w = **W**(D, σ, k). 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.

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 that is. The probability for the sender to cheat is therefore $_2{}^1{}_N$.

Remark. It would also be possible to include the signed statement σ in every single part of the document, but as the maximum size of a watermark is limited by the document's size, this might be problematic for scenarios where the parts are small. Therefore, we embed σ in the complete original document and only embed single bits to the (possibly small) parts of the document.

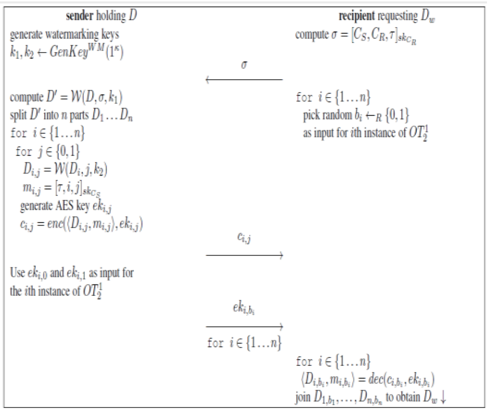
sender holding D
recipient requesting D_w
compute $\sigma = [C_S, C_R, \tau]_{skC_R}$
generate watermarking keys
$k_1, k_2 \leftarrow GenKey^{WM}(1^\kappa)$
σ
compute $D' = W(D, \sigma, k_1)$
split D' into n parts $D_1 \dots D_n$
for $i \in \{1 \dots n\}$
for $j \in \{0,1\}$
$D_{i,j} = W(D_i, j, k_2)$
$m_{i,j} = [\tau, i, j]_{skC_S}$
generate AES key $ek_{i,j}$
$c_{i,j} = enc(\langle D_{i,j}, m_{i,j}\rangle, ek_{i,j})$
for $i \in \{1 \dots n\}$
pick random $b_i \leftarrow_R \{0,1\}$
as input for ith instance of OT_2^1
$c_{i,j}$
Use $ek_{i,0}$ and $ek_{i,1}$ as input for
the ith instance of OT_2^1
ek_{i,b_i}
for $i \in \{1 \dots n\}$
for $i \in \{1 \dots n\}$
$\langle D_{i,b_i}, m_{i,b_i}\rangle = dec(c_{i,b_i}, ek_{i,b_i})$
join $D_{1,b_1}, \dots, D_{n,b_n}$ to obtain $D_w \downarrow$

Figure 3: Protocol for untrusted senders: The sender splits a watermarked document into n parts and creates two different versions of each part by embedding another watermark. The recipient via OT receives one of two versions of each part as well as a signed statement as proof of his choice. The recipient joins the individual parts to create his version of the document.

Timestamp τ to uniquely identify a specific transfer between two parties, and thus assume that no two transfers between the same two parties take place at the same time. However, it would be possible to use a counter that is incremented on each transfer to allow multiple transfers at the exact same time.

Analysis of the Protocol

We now show that the protocol presented in Fig. 3 fulfills the required properties of correctness, no framing and no denial as presented in Section 2.2 :

1) **Correctness:** Assume that both parties follow the protocol steps correctly. Assuming the correctness of the encryption, watermarking, signature and oblivious transfer scheme, we show that for all possible scenarios the guilty party can be determined correctly:

a) ***the sender publishes* D *or* D′:** The auditor does not detect

σ (in the case of D) or the b_i values (in the case of D′) and correctly blames the sender, because both watermarks have to be present in order to blame the recipient.

b) ***the recipient publishes* D_w:** The auditor successfully detects σ and b′ in the leaked document and verifies that is of the correct form. The recipient is able to provide the proof of his choice of b; the auditor verifies b′ = b and suspects the recipient. As there are no further watermarks embedded, the auditor correctly blames the recipient.

False positives in the watermark detection (i.e., a watermark is detected, although it is actually not present) is not a big issue, as the probability that the correct bit string of length n is spuriously detected is negligible. False negatives (i.e., a watermark is not detected, although it is embedded in the document) can be problematic, because if watermarks are not detected the auditor blames the sender. Nevertheless, as included timestamp τ is the same, too. As the auditor asks the recipient to prove his choice of b for this τ , the recipient is able to provide a correct proof, as a valid transfer with timestamp τ actually happened. Analogous to the previous case, the sender can only chose $b^* \in \{0, 1\}^n$ randomly and therefore he can only succeed with negligible probability.

From these two steps it follows that the sender is not able to frame a recipient.

The recipient could also create a watermarked version with a different bit string embedded, if he is able to get $D_{i,1-b_I}$ for some i, but this is only possible if he breaks the OT_2^1 scheme or the encryption scheme, which is only possible with negligible probability. We now show that a recipient cannot cheat during the auditing process, when he proves which version of the document he asked for during the transfer protocol: In order to prove another choice of b he would again have to break the OT_2^1 scheme or the encryption scheme in order to learn the $m_{i,1-b_I}$ for some i or he would need to forge the sender's signature to create $m_{i,1-b_I}$ for some i. As all of this is only possible with negligible probability, the overall probability for the recipient to succeed is negligible. From these two steps it follows that the recipient is not able to publish a document without being caught.

6. Implementation and Microbenchmarking

We implemented the protocol in Fig. 3 as a proof-of-concept and to analyze its performance. For the oblivious transfer sub protocol we implemented the protocol by Naor and Pinkas using the PBC library, which itself makes use of the GMP library. For signatures we implemented the BLS scheme, also using the PBC library. For symmetric encryption we used an implementation of AES from the Crypto++ library. For watermarking we used an implementation of the Cox algorithm for robust image watermarking from Peter Meerwald's watermarking toolbox. We set the α-factor, which determines the strength of the watermark, to a value of 0.1.

We executed the experiment with different parameters to analyze the performance. The sender and recipient part of the protocol are both executed in the same program, i.e., we do not analyze network sending, 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. We measured execution times for different phases of the protocol: watermarking, signature creation, encryption, oblivious transfer and detection. We executed each experiment 250 times and determined the average computation time and the standard deviation.

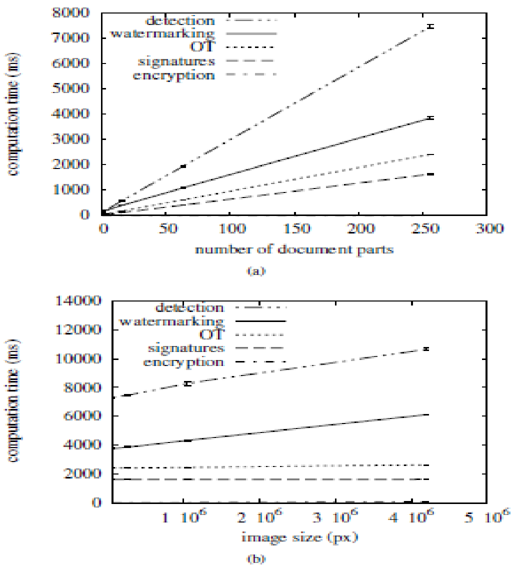

Fig. 4. (a) shows computation times for different numbers of document parts; (b) shows computation times for different image sizes.

In the first experiment we used an image of size 512 × 512 pixels and changed the number of parts the image was split into. We show the results in Fig. 4(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.

We find that latencies of a few seconds are acceptable in the scenarios that we considered. Additionally, as we show in our experiments in Fig. 4(a), one can easily perform a tradeoff between performance and security. It is also possible to use the OT extension technique presented in [20] to increase the efficiency of oblivious transfer. Although we use only image files as documents in our experimental implementation, we stress that the same mechanism can be used for all types of data for which robust watermarking schemes exist.

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 $\frac{s}{n}$ 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 our case the one by Naor and Pinkas [15]). 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 in our case) (64 bytes). In total the sender sends 2s + 218n bytes and the recipient sends 32n + 48 bytes. For our example with an image of

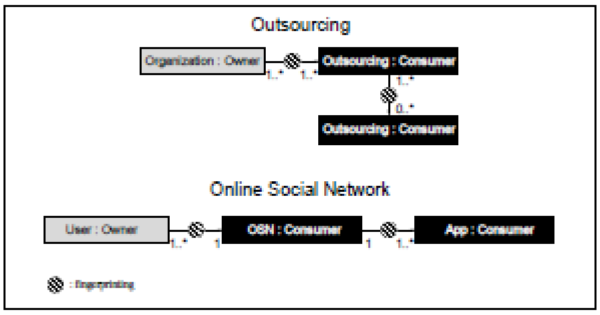

Fig. 6. Outsourcing Scenario

size s = 1MB and n = 64 we have a communication overhead of 2.008MB for the sender and 2.05KB for the recipient, which we find to be practical.

The auditor to one of the outsourcing companies. This outsourcing company can in turn reveal additional fingerprints 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. In the example given in the introduction, there were three outsourcing companies involved and a data leakage could not be clearly associated with one of these. The responsible party can be clearly found using LIME.

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 our implementation the sender also needs to store the original document.

Multiple Iterations

Fig. 5(a2) shows an image that was transferred

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 our 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 outsourcing company itself can outsource tasks to other outsourcing companies and thus relay the documents, again using fingerprinting. It is important to notice that a single organization can outsource to may 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 second diagram in Fig. 6 shows an online social networking scenario. The users of the network are the owners, as they enter their personal information, post messages, etc. The online social network (OSN) 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.

Collusion Resistance

The collusion resistance of our scheme depends on the collusion resistance of the underlying watermarking scheme. Assume several consumers are working together in order to create an untraceable version of a document. Then their approach is to merge the versions they rightfully obtained to create a new version where the watermarks cannot be detected.

As the detection of σ is just a detection of a watermark in the complete document, we obviously have the same collusion resistance as the watermarking scheme for this case. The case of the detection of a bit b_i in a part D_i is again just a detection of a watermark, so the collusion resistance is again the same as for the watermarking scheme. However, we have to know which detected bit belongs to which consumer; so that we can still guarantee that the sender cannot frame the receiving consumers. Linking the detected bits to the responsible consumers is possible, as for each consumer a different embedding key was used. As for each part multiple bits might be detectable, the probability for a sender to successfully frame the receiving consumers is less than or equal to the probability of framing a single recipient successfully, as he still would have to guess all the bits correctly. However, we have to note that in order to successfully mount collusion attack against our scheme, it is sufficient to mount a collusion attack against 1 of the n + 1 watermarks that are used, where n is the number of parts the document was split into.
We can conclude that our scheme tolerates collusions to a certain extent, when it is used with a collusion resistant watermark, without losing its key properties.

Error Tolerance

Depending on the quality of the underlying watermarking scheme, it may be too strong to require that all bits b_i are detected correctly. Therefore, it could be a good idea to introduce some error tolerance. However, we have to keep in mind that this will increase the probability of the sender successfully framing an innocent recipient. There are two different kinds of errors that can occur: the first one is that no bit can be detected, and the second one is that a wrong bit is detected. Assume the document is split into n parts. Tolerating a non-detectable bit increases the probability of successful framing by a factor of 2. Instead of guessing a bit string $b \in \{0, 1\}^n$, it is sufficient to guess $\underline{b} \in \{0, 1\}^{n-1}$. Tolerating a wrong bit is worse, as it increases this probability by a factor of (n + 1). Instead of accepting just the correct bit string, we also accept all bi strings that are changed at exactly one position. As there are n positions, we additionally accept n bit strings; hence the number of accepted bit strings and thus the probability of guessing one of these is higher by a factor of:

n + 1. If we want to allow some error tolerance while keeping the probability of successful framing to be small, we

have to choose a larger n; e.g., to tolerate 128 non-detectable bits, we choose
n = 256 and have the same framing probability as with n = 128 and no tolerance.

Possible Data Distortion
In our experiment, we used a simple splitting algorithm: We split the image into n equally sized squares. However, when we used a strong watermark for the small parts (that is the α-factor used by the Cox algorithm is 0.5), differences between adjacent parts became visible even though the single watermarks are imperceptible. The resulting image can be seen in Fig. 5(b1). This effect becomes even stronger after multiple iterations as observed in Fig.5(b2). In some cases, this distortion might affect the usability of the document. We stress however, that we were still able to obtain good results with our approach. In

7. Related Work

A preliminary shorter version of this paper appeared at the STM workshop. This version constitutes a significant extension by including the following contributions: We give a more detailed description of our model, a formal specification of the used primitives, an analysis of the introduced protocol, a discussion of implementation results, an application of our framework to example scenarios, a discussion of additional features and an extended discussion of related work.

Other Models
Hasan, Sion and Winslett present 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.

The model introduced in intends to help the data distributor to identify the malicious agent which 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. In LIME the relationship of data distributor and agents corresponds to the relationship between data owner and consumer and the model could be used as an alternative method to trace the information given to the consumers.

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. In the authors present 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. As presented in the paper, this problem can be solved by the usage of watermarking and possibly even by using complete fingerprinting protocols during the generating phase of the document.

Other Fingerprinting Protocols
In Poh 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. All presented schemes use watermarking to trace the guilty party and most presented protocols make use of *watermarking in the encrypted domain*, where encrypted watermarks are embedded in encrypted documents. A major advantage of our scheme is that it can be used with every existing watermarking scheme without any modification. The schemes relying on watermarking in the encrypted domain only work with watermarking schemes that are designed for this technique. A new scheme presented is based on chameleon encryption. In Sadeghi also examines several fingerprinting schemes and presents new constructions for symmetric, asymmetric and anonymous fingerprinting schemes. The asymmetric scheme uses a homomorphism commitment scheme to compute the fingerprinted version of the document.

Domingo-Ferrer presents the first fingerprinting protocol that makes use of oblivious transfer in. In the scheme, documents are split into smaller parts and for each part two different versions are created. Then the recipient receives one version of each part via oblivious transfer and in return sends a commitment on the received part. The recipient can now be identified by the unique combinations of versions he received. The protocol has several flaws, as discussed. The main problem is that a malicious sender can offer the same version twice in the oblivious transfer, so that he will know which version the recipient receives.

Sadeghi and Hanaoka et al. propose different solutions; the former lets the sender open some pairs to validate that they are not equal and the latter uses oblivious transfer with a two-lock cryptosystem where the recipient can compare both versions in encrypted form. However, both proposed solutions have some flaws themselves. The problem is that it is possible to create two different versions with the same watermark, so even if the equality test fails, the two offered versions can still have the same watermark and the sender will know which watermark the recipient received. Also, the fix proposed in ruins the negligible probability of failure, as it does not split the document into parts, but creates n different versions and sends them via 1-out-of-n oblivious transfer.

Domingo-Ferrer presents another protocol based on oblivious transfer, but again the sender can cheat during oblivious transfer. Presents another protocol using oblivious transfer. The protocol uses an approach similar to the chameleon encryption, and using 1-out-of-n oblivious transfer a decryption key is transmitted so that the sender does not know it. The protocol suffers from the same problems as the one presented in; namely, the sender can guess the key used by the recipient with non-negligible probability $\frac{1}{n}$ and the sender can even cheat in the oblivious transfer by offering the same key n times, so that he will know the key used by the recipient.

We see that all asymmetric fingerprinting protocols based on oblivious transfer that have been proposed so far suffer from the same weakness. We circumvent this problem in our protocol by additionally sending a signed message including the watermark's content, so that the recipient is able to prove what he asked for. In contrast to the watermark, this message can be read by the recipient, so he can notice if the sender cheats.

Broadcasting

Parviainen and Parnes present an approach for distributing data in a multicast system, so that every recipient holds a differently watermarked version. The sender splits the file into blocks and for each block he creates two different versions by water-marking them with different watermarks and encrypting them with different keys. Each recipient is assigned a set of keys, so that he can decrypt exactly one version of each part. The resulting combination of parts can uniquely identify the recipient. In Adelsbach, Huber and Sadeghi show another approach for a broadcasting system that allows identification of recipients by their received files. With a technique called *fingercasting*, recipients automatically embed a watermark in files during the decryption process. The process is based on the chameleon cipher [29], which allows one to decrypt an encrypted file with different decryption keys, to introduce some noise that can be used as a means of identification. In [38] Katzenbeisser et al. use the technique of fingercasting together with a randomized fingerprinting code in order to provide better security against colluding attackers. However, in these broadcasting approaches the problem of an untrusted sender is not addressed.

Watermarking

LIME can be used with any type of data for which watermarking schemes exist. Therefore, we briefly describe different water-marking techniques for different data types. Most watermarking schemes are designed for multimedia files such as images, videos, and audio files. In these multimedia files, water-marks 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 us 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.

Suchanek et al. propose an interesting approach for watermarking ontologies. In this approach the authors propose to rather remove existing information than adding new information or modifying existing information. Thereby the wa-termarking scheme guarantees that no false entries are introduced. The above schemes can be employed in our framework to create data lineage for documents of the respective formats. The only modification that might be necessary when applying our 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 tech-niques available. In they discuss multiple re-watermarking and in the focus is on segmented watermarking. Both papers show in experimental results that multiple watermark-ing is possible which is very important for our scheme, as it allows us 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. In Sadeghi 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 in also hides the content of the watermark itself and are therefore unfit for our case, 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.

8. Conclusion and Future Directions

We present LIME, a model for accountable data transfer across multiple entities. We define participating parties, their inter-relationships and give a concrete instantiation for a data transfer protocol using a novel combination of oblivious transfer, robust watermarking and digital signatures. We prove its correctness and show that it is realizable by giving micro benchmarking results. By presenting a general applicable framework, we introduce accountability 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. LIME is flexible as we differentiate between trusted senders (usually owners) and untrusted senders (usually consumers). 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 (e.g. a judge).

Our 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.

References

[1] "Chronology of data breaches," http://www.privacyrights.org/data-breach.
[2] "Data breach cost," http: //www.symantec.com/about/news/release/article.jsp?prid=20110308 01.
[3] "Privacy rights clearinghouse," http://www.privacyrights.org.
[4] "Electronic Privacy Information Center (EPIC)," http://epic.org, 1994.
[5] "Facebook in Privacy Breach," http://online.wsj.com/article/ SB10001424052702304772804575558484075236968.html.
[6] "Offshore outsourcing," http://www.computerworld.com/s/article/ 109938/Offshore outsourcing cited in Florida data leak.
[7] A. Mascher-Kampfer, H. St ogner,¨ and A. Uhl, "Multiple re-watermarking scenarios," in *Proceedings of the 13th International Conference on Systems, Signals, and Image Processing (IWSSIP 2006)*. Citeseer, 2006, pp. 53–56.
[8] P. Papadimitriou and H. Garcia-Molina, "Data leakage detection,"
[9] "Pairing-Based Cryptography Library (PBC)," http://crypto.stanford.edu/pbc.
[10] I. J. Cox, J. Kilian, F. T. Leighton, and T. Shamoon, "Secure spread spectrum watermarking for multimedia," *Image Processing, IEEE Transactions on*, vol. 6, no. 12, pp. 1673–1687, 1997.
[11] B. Pfitzmann and M. Waidner, "Asymmetric fingerprinting for larger collusions," in *Proceedings of the 4th ACM conference on Computer and communications security*, ser. CCS '97, 1997, pp. 151–160.
[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] A. Adelsbach, S. Katzenbeisser, and A.-R. Sadeghi, "A computational model for watermark robustness," in *Information Hiding*. Springer, 2007, pp. 145–160.
[14] J. Kilian, F. T. Leighton, L. R. Matheson, T. G. Shamoon, R. E. Tarjan, and F. Zane, "Resistance of digital watermarks to collusive attacks," in