**Data Lineage in Malicious Environments**

**Abstract:**

Intentional or unintentional leakage of confidential data is undoubtedly one of the most severe security threats that organizations face in the digital era. The threat now extends to our personal lives: a plethora of personal information is available to social networks and smart phone providers and is indirectly transferred to untrustworthy third party and fourth party applications. In this work, we present a generic data lineage framework LIME for data flow across multiple entities that take two characteristic, principal roles (i.e., owner and consumer). We define the exact security guarantees required by such a data lineage mechanism toward identification of a guilty entity, and identify the simplifying non-repudiation and honesty assumptions. We then develop and analyze a novel accountable data transfer protocol between two entities within a malicious environment by building upon oblivious transfer, robust watermarking, and signature primitives. Finally, we perform an experimental evaluation to demonstrate the practicality of our protocol and apply our framework to the important data leakage scenarios of data outsourcing and social networks. In general, we consider LIME , our lineage framework for data transfer, to be an key step towards achieving accountability by design.

**Existing system:**

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. 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 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.

**Disadvantages:**

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.

**Proposed System:**

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. 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.

**Advantage:**

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.

**Implentation Modules:**

**1.Microbenchmarking**

**2.Possible Data Distortion**

**3.Broadcasting**

**4.Watermarking**

**Micro benchmarking:**

We implemented the protocol in as a proof-of-concept and to analyze its performance. For the oblivious transfer sub protocol we implemented the protocol] 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 We set the -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.

**Possible Data Distortion:**

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   
- 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 . This effect becomes even stronger after multiple iterations as observed . 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 Fig. 5(a1) we used the Cox algorithm with an alpha factor of 0.1 and no distortion is visible. It might be interesting to investigate if this problem can be circumvented by using more elaborate splitting algorithms. As most watermarking schemes make use of the contiguity of information in the document, this is not a trivial task.

**Broadcasting:**

We 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 watermarking 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. It show another approach for a broadcasting system that allows identification of recipients by their received files. With a technique called finger casting, recipients automatically embed a watermark in files during the decryption process. The process is based on the chameleon cipher , 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. It use the technique of finger casting 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:**

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. 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.

# **Configuration:-**

# **H/W System Configuration:-**

# System - Pentium –IV 2.4 GHz

Speed - 1.1 Ghz

RAM - 256MB(min)

Hard Disk - 40 GB

Key Board - Standard Windows Keyboard

Mouse - Logitech

Monitor - 15 VGA Color.

# 

# **S/W System Configuration:-**

* Operating System :Windows/XP/7.
* Application Server : Tomcat5.0/6.X
* Front End : HTML, Java, Jsp
* Scripts : JavaScript.
* Server side Script : Java Server Pages.
* Database : Mysql 5.0
* Database Connectivity : JDBC.