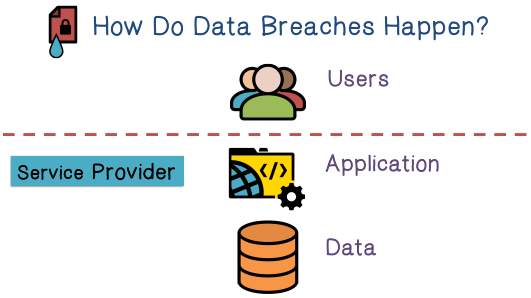
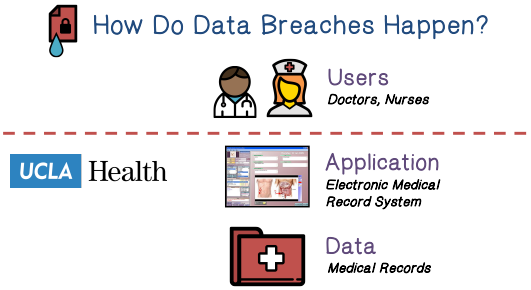
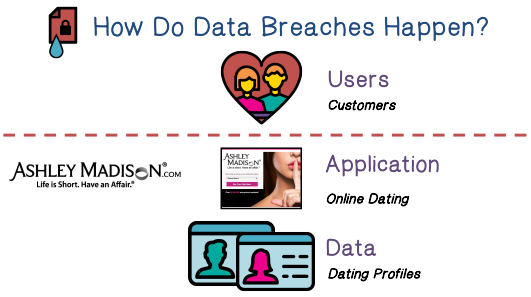
In this lesson, we continue our focus on The Cloud. What a concept of shared sources is an excellent idea. It is also rabbit hole of security issues. We will examine and analyze a number of Cloud oriented attacks. In particular, we discussed attacks on data privacy and the challenges in data privacy protection.



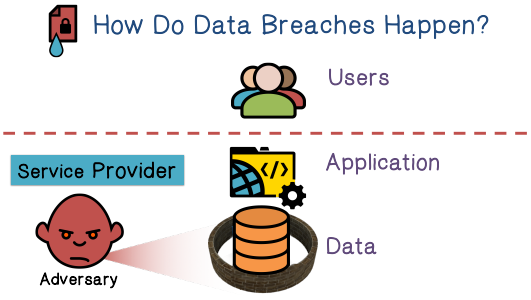
Let's discuss data privacy issues in a cloud environment. First let's review how data breaches happen. In a cloud environment the users use applications or data stored in a cloud provider.



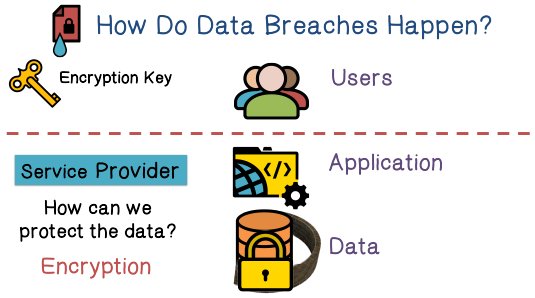
For example, the users can be doctors and nurses, and the application can be the electronic medical record system. And the data obviously is the medical records. And the called EMR here can be a private called such as UCLA Health.



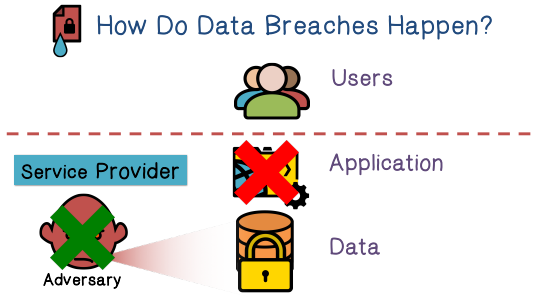
As another example, the users can be average consumers. And the application can online dating. And of course, the data can be the dating profiles.



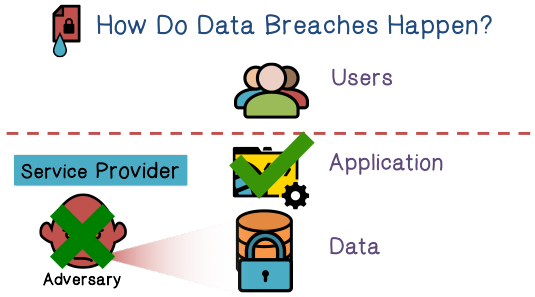
In most cases, including the examples that we just discuss, data is the most valuable target to the attacker. For example, the attacker may want to steal the data.



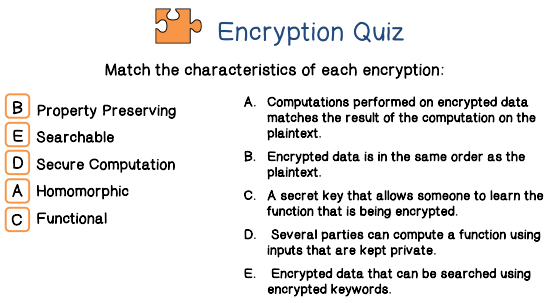
Now the question is, how do we protect the data? Obviously, we can encrypt the data. For example, the users can have their own secret encryption key for their data.



And once the data is encrypted the adversary can no longer read the data. The problem when the data is encrypted the applications can no longer use the data as it is. For example, if the application is a word processing program such as Microsoft Word, it cannot process encrypted data directly.



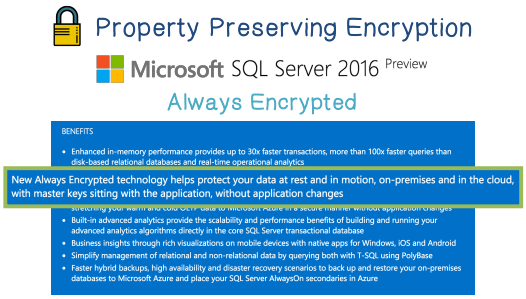
Therefore, the real question is can we protect the data while we also let the application work? In other words, we want to protect the data by encrypting the data, so that the adversary cannot read our data. On the other hand, we also want the application to continue to work.



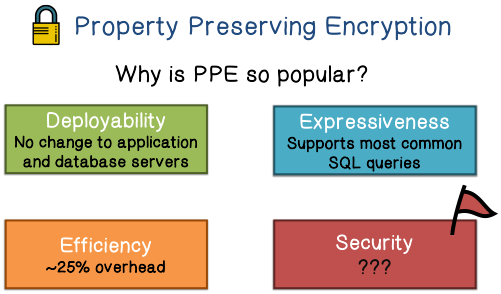
Before we go on, let's do a quiz on encryption. Match the characteristics of each encryption.

In property preserving encryption, some selective properties of the alternate data are preserved, such as the order. Searchable encryption means that the encrypted data can be searched using the encrypted keywords. In secure computation, multiple parties can compute a function using inputs that are kept private. In Homomorphic encryption, computations performed directly on encrypted data, have the same result as the computations on the plaintext. In functional encryption, the possession of a secret key will allow someone to learn the function that is being encrypted.

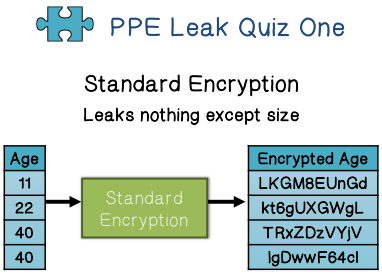
Property Preserving Encryption is one way to protect data while allowing application to continue to work. And this approach is widely deployed in various environments including the Cloud.



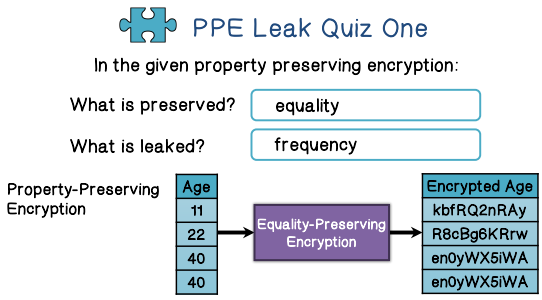
For example, Microsoft advertises that data is always encrypted. On the other hand, the applications continue to work.



Property Preserving Encryption has several advantages. There's no need to change application and database servers, it supports common data retrieval methods, including SQL queries. It is reasonably efficient, but what about security? Although it widely believed that it is secure, we should take a closer look.

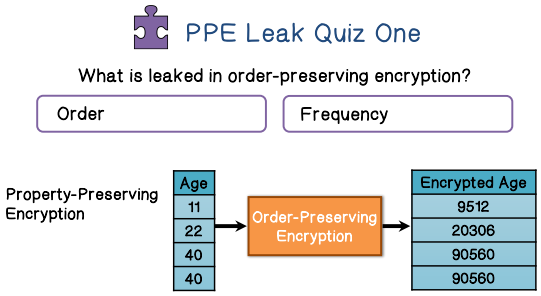


To understand the potential security issues of Property Preserving Encryption, let's do a quiz. First, let's review standard encryption. In standard encryption, there is no preserving of properties. For example, suppose we encrypt the age information. The encrypted data leaks nothing except the size of the original plain text data. For example, we know that there are four entries.



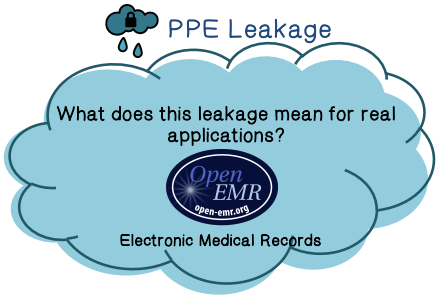
Now let's take a look at an example of property preserving encryption. Again, we want to encrypt the age information, but we preserve equality. Therefore in encrypted data, we see that these two values are the same, because the original paying tax values are the same. So what is preserved?

As we said, equality is preserved, but what is leaked is a frequency. Because we now know that there's one value that appears twice.

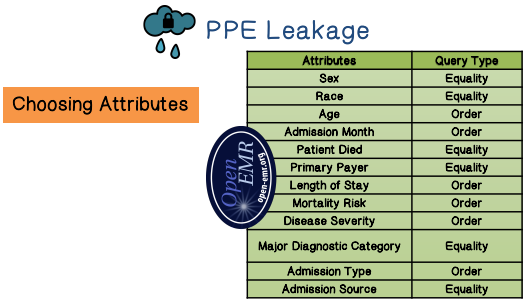


Now let's take a look at another example of property-preserving encryption. Again, we want to encrypt the age information. And this time, we preserve the order. As we can see, in the encrypted data, the order of the original values is preserved. The question is, what is leaked in order-preserving encryption?

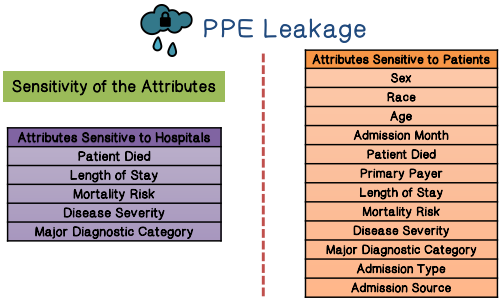
Obviously, the order is leaked. Frequency is also leaked because the same value in plain text will be the same value in encrypted text.



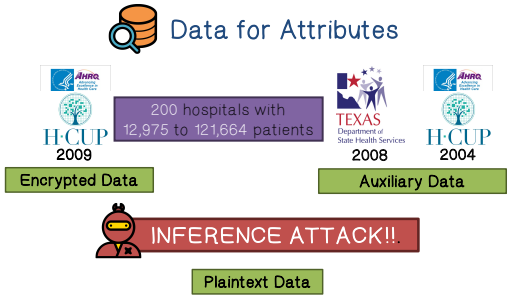
The previous examples are simple. Now the question is, what does the leakage in property preserving encryption really mean for real applications? We can take a look at the electronic medical records.



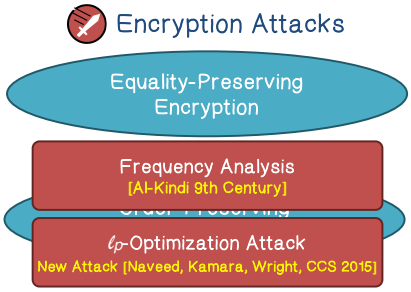
Here are some attributes in electronic medical records. These attributes are typically used in equality queries or ordering. In other words, to ensure that the applications will continue to work, these attributes will be encrypted to preserve equality or order.



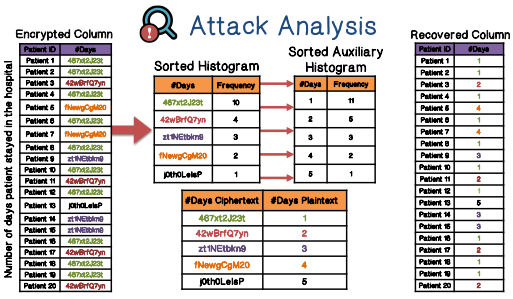
These attributes are sensitive to either the hospital or the patient or both.



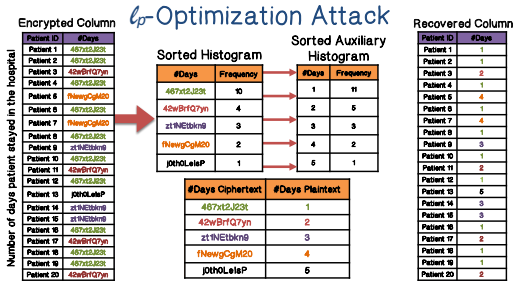
The data as we can obtain from the hospitals is encrypted. On the other hand, there's information, or auxiliary data that is public. We will show that using both the encrypted data and auxiliary data, an attacker can launch inference attack to obtain plaintext data.



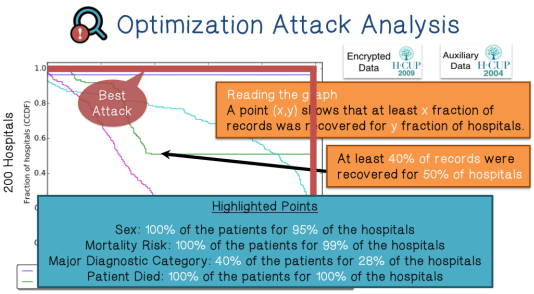
Recall that these attributes are encrypted using either Equality-Preserving Encryption or Order-Preserving Encryption. An attacker can use frequency analysis to defeat Equality-Preserving Encryption. And this attack can be further optimized.



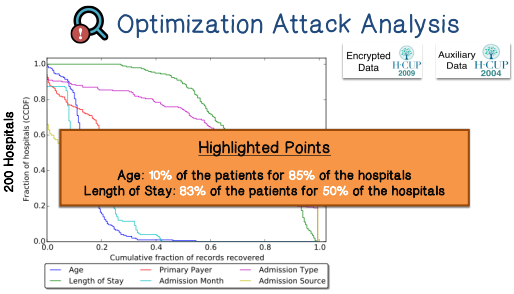
Let's use an example to illustrate frequency analysis attack. Here is encrypted data of the number of days a patient stays in hospital. As you can see the data is encrypted to perform frequency analysis we first sort the data and record the frequency. We also sort and record the frequency of the auxiliary data. For example, there's public information of how frequent a patient will stay for one day versus two days and so on. By matching these two histograms we can link a cyber text value to a plain text value. For example, for the first cyber attack value the frequency is ten. And we know that from the auxiliary data, the plain text value one has frequency of 11. Therefore, we can link the ciphertext value to plaintext value one. And therefore, with frequency analysis, we can uncover the plaintext data without the encryption key.



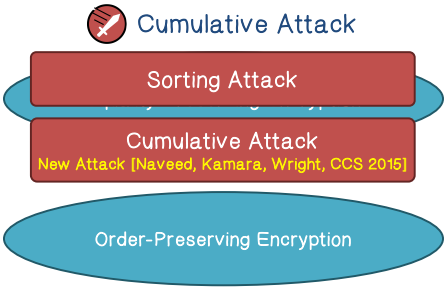
This attack can be generalized and optimized. Again, start with the encrypted data, we have obtained a frequency histogram. We also obtain a frequency histogram from the auxiliary data. The basic idea is then, to find an assignment from servertext to plaintext that minimizes a gives cost function. Here the cost function is the distance between the histograms. This has the effect of minimizing the total mismatch in frequencies across all plaintext-ciphertext pairs. For example, for this assignment, the cost is this. Whereas the assignment on mapping, it has the minimum cost is this. There's an algorithm that can find the assignments that has the minimum cost. With this algorithm, we can find the assignment from ciphertext to plain text and then decipher the original encrypted data.



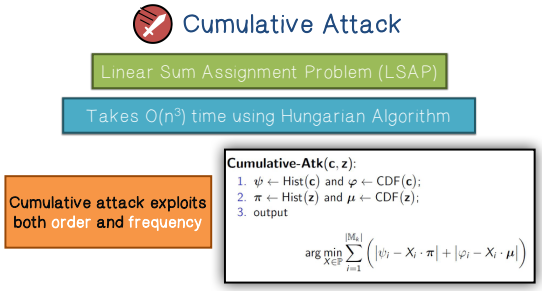
Here are some results of applying this attack on the electronic Medical records. The x axis is the cumulative fraction of records recovered, and the y axis is the fraction of hospitals. That is, in all of this parts, our point xy shows that at least x fraction of records was recovered for y fraction of hospitals. For example, for disease severity at least 40% of the records were recovered for 50% of the hospitals. The best result we can obtain is that all records, for all hospitals, were recovered. Here are the highlights of the results. For example, for sex, 100% of the patients for 95% of the hospitals recovered. And for major diagnostic category, 40% of the patients for 28% of the hospitals were recovered.



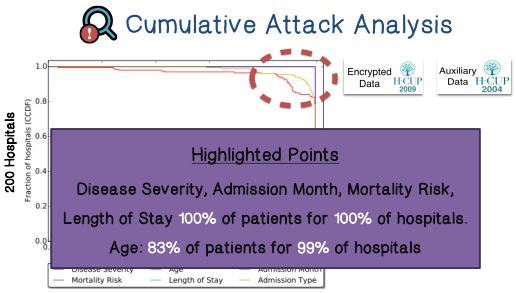
And we show more results here the highlights are, for age 10% of the patients for 85% of the hospitals were recovered. For length of stay, 83% of the patients for 50% of the hospitals were recovered.



We have discussed on equality preserving inscription. Now, let's discuss a tax on order preserving inscription. We can use sorting attack, but a more effective attack is cumulative attack.



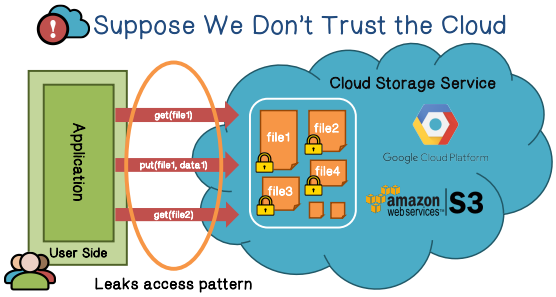
Given a data column that has been encrypted using order preserving inscription. An adversary can learn not only the frequencies, but also the relative ordering of the encrypted values. Combining ordering with frequencies, the attacker can tell for each server text C, what fraction of the encrypted values are less than C. More formally, this is known as the empirical cumulative distribution function of the dataset or CDF. In the cumulative attack, an attacker leveraged the CDF to improve the ability to match plaintext to ciphertext. Intuitively, if a given ciphertext is greater than 90% of the ciphertext in the encrypted data, then we shall match it to a plaintext that is greater than about 90% of the auxiliary data. This problem belongs to a category of Linear Sum Assignment Problem. Therefore, we can use an algorithm to find the mapping of plaintexts to ciphertext that minimizes the total sum of mismatch in frequency plus the mismatch in CDFs across all plaintext, ciphertext pairs.



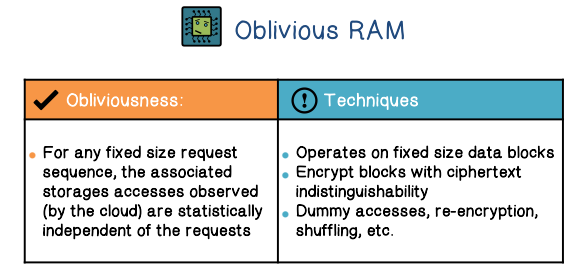
Here are some results of this attack and the highlights are for disease severity, admission month, mortality risk, length of stay 100% of the patients for 100% of the hospitals were recovered. And for age, 83% of the patients for 99% of the hospitals were recovered.

Attribute	Equality-Preserving Encryption Attack	Order-Preserving Encryption Attack
Sex	100% patients, 85% hospitals	–
Race	60% patients, 70% hospitals	–
Age	10% patients, 65% hospitals	83% patients, 99% hospitals
Admission Month	20% patients, 55% hospitals	100% patients, 100% hospitals
Patient Died	100% patients, 100% hospitals	–
Primary Payer	90% patients, 38% hospitals	–
Length of Stay	83% patients, 50% hospitals	100% patients, 100% hospitals
Mortality Risk	100% patients, 99% hospitals	100% patients, 100% hospitals
Disease Severity	100% patients, 51% hospitals	100% patients, 100% hospitals
Major Diagnostic Category	40% patients, 28% hospitals	–
Admission Type	60% patients, 65% hospitals	79% patients, 100% hospitals
Admission Source	90% patients, 38% hospitals	–

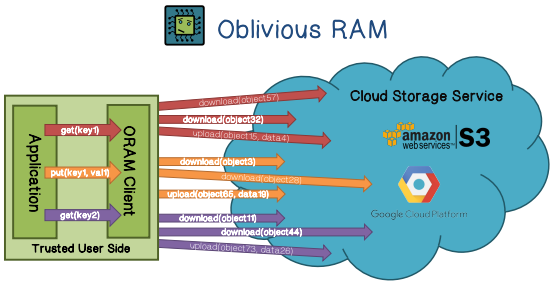
Here is a summary of the results of attacks on electronic medical records. As you can see, the confidentiality of many attributes is compromised.



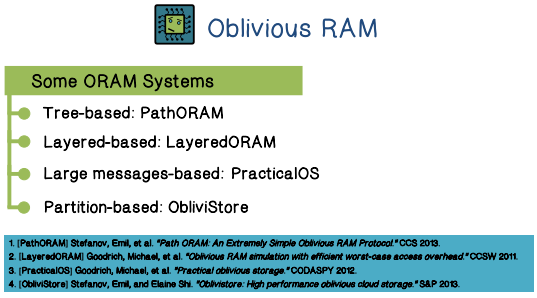
Now let's discuss another data privacy issue in the cloud environment. Suppose we don't trust the cloud provider? We can either encrypt our data on the cloud storage server and keep the keys to ourself. When we need to use the data, we can fetch the encrypted data to our environment, encrypt the data, then use the application in our local environment. The question is, is this efficient? No, because the data access patterns can still leak information, such as what kind of computing tasks are being performed.



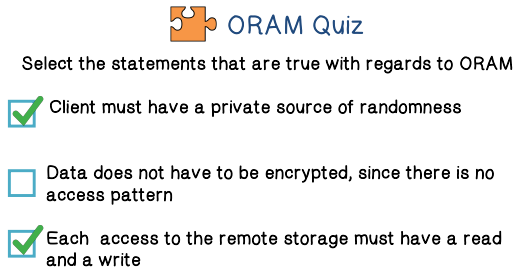
A promising approach to eliminate this kind of leakage is to use oblivious RAM, or ORAM. ORAM can hide access patterns. The main idea is that using ORAM, while the cloud provider can still observe data access, the access patterns are independent of the actual data requests. Some of the main techniques include O data access operating on fixed size data blocks. The data is encrypted, not using property preserving encryption. In addition, ORAM also use dummy accesses, or re-encrypt data and shuffle data around.



Here's a high level pictorial example of ORAM at work. The application needs to access confidential data such as getting an encryption key, encrypt and store data, and getting another encryption key. For each of these access requests, there are multiple requests to the cloud server. These requests all fixed sized data objects, and they are both read and writes. That is, regardless whether the original request is for read or write, the actual dummy accesses include both read and writes. And the data objects are all of fixed size and not distinguishable from each other. And this is how you hide the access pattern from the cloud provider.



ORAM is an active research area, and if you are interested, here are some papers.



Now, let's do a quiz on ORAM. Select the statements that are true with regards to ORAM.

The first statement, client must have a private source of randomness. This is true, because the ORAM client must generate random access patterns. The second statement, data does not have to be encrypted, since there's no access pattern. This is obviously wrong, because we want to protect data from the cloud provider. Therefore, encryption is actually the first requirement. The third statement each access to the remote storage must have a read and a write, this is correct. Because we want to hide from core providers the fact that we only reading or writing data, therefore we will include dummy reads and writes.