On The Limits of Privacy Provided by Order- Preserving Encryption

Vladimir Kolesnikov and Abdullatif Shikfa

Much of the value of cloud services lies in leveraging client data, which often conflicts with the client's desire to keep that data private. Reconciling these contradictory requirements is an important research and engineering problem, whose efficient solution would have a far-reaching business impact. Generic theoretical approaches, such as fully-homomorphic encryption, are inefficient. Ad hoc approaches, such as order-preserving encryption (OPE), provide solutions to a limited class of problems (e.g., evaluating encrypted range queries). Security achieved in real systems, even if an "ideal OPE" is employed, is hard to evaluate, and is often only illusory, since the ability to order ciphertexts may reveal a lot about the underlying plaintexts. We concentrate on a typical application of OPE, encrypted searchable webmail service. We describe how the use of OPE in this setting may divulge information and discuss approaches to minimize its impact. The main avenue to improve privacy is to appropriately limit the type of interactions that should be allowed with a webmail server. © 2012 Alcatel-Lucent.

Introduction

The value of many cloud services lies in the ability to leverage a client's data. Typical examples include data storage, webmail service, advertising, and geolocation services. However, the need to access a client's data (e.g., files, email, location) is often at odds with the need to protect the privacy of client data, which requires encryption for data protection. Reconciling these contradictory requirements, and achieving computation on encrypted data, is an important research and engineering problem whose efficient solution would have a far-reaching business impact.

A breakthrough theoretical approach, fully-homomorphic encryption (FHE), described in [6] and a number of follow-up works, while generic, is currently inefficient, and seems unlikely to become truly practical in the foreseeable future. Although a significant effort is underway in the theoretical community to improve the performance of FHE, it is unlikely that fully-homomorphic encryption will approach the efficiency of current public key encryption (PKE) schemes any time soon. Intuitively, this is because a fully-homomorphic cryptosystem must provide the same strong security guarantees as PKE, while supporting the additional algebraic structure to allow for homomorphic operations. The extra structure weakens security and leads to a need for countermeasures, which are implemented at the cost of performance. Further, even performance equivalent to that of a "regular" public-key encryption scheme such as RSA is

Bell Labs Technical Journal 17(3), 135–146 (2012) © 2012 Alcatel-Lucent. Published by Wiley Periodicals, Inc.
Published online in Wiley Online Library (wileyonlinelibrary.com) • DOI: 10.1002/bltj.21564

unsatisfactory in most scenarios. This is because the generic approach of computing under encryption requires performing a public-key operation for each trivial step of the client-server computation, such as addition.

Similarly, a generic approach based on the garbled circuit [7, 13] is also inapplicable in a number of cloud settings, where the computed functions are large (measured as a number of gates in a boolean circuit computing the function). Despite relying on symmetric encryption, and hence being dramatically more efficient than FHE, the garbled circuit approach also suffers from overhead, both in computation and communication, linear to the size of the computed function.

On the other hand, ad hoc approaches such as the order-preserving encryption (OPE) described by Aggrawal et al. [1], provide solutions to a limited class of computations on encrypted data, such as evaluating encrypted range queries. (Recall, an OPE has the property that for any two messages m_1 and m_2, where $m_1 < m_2$, it holds that $OPE(m_1) < OPE(m_2)$. Further, these ad hoc approaches often lack the usual cryptographic formalization and analysis. One notable exception is the OPE scheme described by Boldyreva et al. [3], which is probably as strong as an ideal OPE object—a pseudorandom order-preserving function. However, it is often not clear what level of security is provided by an application using this ideal OPE as Boldyreva et al. warn of possible security compromises.

Indeed, in contrast with "regular" encryption schemes, OPE ciphertexts by necessity reveal a significant amount of information about the plaintext they encrypt. For example, the magnitude of the OPE ciphertext can allow an adversary to develop an estimate of the range of the corresponding plaintext. Worse, the estimate can be made more precise, given additional ciphertexts. Worse yet, auxiliary information available to the adversary, such as plaintext/ciphertext pairs (especially if plaintexts are chosen by the adversary), and knowledge of plaintext distribution can allow them to refine these estimates and eventually (actually, quite quickly) decrypt protected information. The type and amount of auxiliary information available varies by application, and is very hard to formalize and analyze in general.

Panel 1. Abbreviations, Acronyms, and Terms

AES—Advanced Encryption Standard
CCA—Chosen-ciphertext attack
CPA—Chosen-plaintext attack
FHE—Fully-homomorphic encryption
KPA—Known-plaintext attack
OAEP—Optimal Asymmetric Encryption Padding
OPE—Order-preserving encryption
PKE—Public key encryption
PKI—Public key infrastructure
POPE—Probabilistic OPE
ROPF—Random order-preserving function
UI—User interface

In this work, we will concentrate on one of the typical applications for OPE—encrypted webmail service, which allows searches and sorting. We will characterize the information leakage inherent to OPE in this setting and discuss approaches to minimize its impact. We believe the best path toward improving privacy is by appropriately limiting the types of interaction the webmail server should be allowed (and hence limiting the information available to the server, including leaked information).

This work is intentionally self-contained and high-level. It was our goal to present most of the issues, analysis, and solutions at a level appropriate for security engineers and system architects.

OPE Applications and Our Setting

There are a number of applications which could benefit from order-preserving encryption. For further evidence of the importance of this problem and the need to examine the security provided by OPE, we list below several natural scenarios. It is clear that the ability to securely compare integers under encryption is a powerful primitive, which facilitates a variety of interesting cloud and security applications.

For instance, Lu et al. [9] propose several techniques to perform privacy-preserving searches on multimedia. In their work, they first extract visual features from the multimedia document. These features are hierarchically clustered and assigned to a

representative feature called "visual word." The entire multimedia document is then represented as a set of keywords which are indexed. For privacy protection the word frequency values are encrypted with OPE, enabling a ranked search on the indexes. Wang et al. [12] propose a scheme that supports secure and efficient ranked keyword searches over encrypted data stored in the cloud by applying order-preserving encryption on certain relevance criteria such as the frequency of keywords. Ding and Klein [5] propose an application-level encryption solution to protect the privacy and confidentiality of health data. In particular, their solution relies on order-preserving encryption to enable some operation on dates expressed in milliseconds without first having to decrypt them. These and other applications of OPE (e.g., [8, 10, and 11]) all target an outsourced computation or storage model, which are key characteristics of cloud computing.

While our discussion largely applies to the above applications as well, in this work we will concentrate on the webmail setting, providing a brief overview here and presenting the issues in more detail in a following section. We consider a webmail server M, which, upon receipt of (possibly encrypted) email m destined to client C, forwards m to C. In turn, C processes the email simply by encrypting it with OPE (we will discuss the encryption process in more detail later), and uploads to M for storage. Now, given his OPE key, C is able to access his email from any web browser. Importantly, he has access to sorting and searching functionality, while preserving a high degree of protection for his data.

Outline of the Paper

We start with preliminaries, describing our setting, the basics of secure computation and the related work. We discuss our main application, webmail, and sketch a natural architecture for its use with encryption. We then identify several critical vulnerabilities of the system which persist even if an ideal OPE implementation is available. To remedy the situation, we describe several simple design principles, adherence to which will greatly improve security in OPE applications. We then provide examples of new and more secure OPE-based webmail architectures.

Order Preserving Encryption

In this section, we informally recall the definition of OPE introduced by Boldyreva et al., and describe the scheme's basic properties and limitations.

An order-preserving symmetric encryption (or OPE) scheme is a deterministic symmetric encryption scheme whose encryption algorithm produces ciphertexts that preserve the numerical ordering of the plaintexts. Order-preserving encryption was introduced by Aggrawal et al. in 2004 [1], and the first formal study of the concept and its security was performed by Boldyreva et al. in 2009 [3].

Let D and R be finite ordered sets (we can consider them to be subsets of natural numbers for the sake of simplicity). We say that that OPE is an order-preserving encryption scheme with plaintext space D, ciphertext space R, and key space K. For any choice of keys $k \in K$ and any choice of inputs $x_1, x_2 \in D$, the following holds:

$$x_1 < x_2 \,3\, OPE(k,x_1) < OPE(k,x_2)$$

We may omit the key k in our notation when it is clear, and write $OPE(x_1)$ to denote $OPE(k,x_1)$.

The security of OPE is defined by comparison to an ideal (though not necessarily efficiently computed or stored) object that satisfies the hiding and order-preservation properties. Boldyreva et al. define this object to be a function, chosen at random from the set of all order-preserving functions mapping D to R. This object is called a random order-preserving function (ROPF). Unlike the well-understood notion of random functions, ROPF is an unstudied and a less intuitive object. In fact, it is not at all clear what information ROPF leaks about underlying data. Boldyreva et al. provide some characterization of the security offered by the ideal ROPF function, which, they warn, is a weak characterization.

One of the most basic (and weakest) cryptographic properties is one-wayness. One-wayness merely requires that a function, evaluated on a *random* value, cannot be inverted by the adversary. However, it allows in particular that most of the pre-image bits can be determined. We note that even the one-wayness property of ROPF was left open in Boldyreva's original paper, and subsequently shown to hold in a

follow-up paper [4]. In the OPE context, Boldyreva et al. [4] introduce the necessary adjustments to the typical definition of one–wayness. In contrast to the traditional definition, the adversary is not allowed to forward-evaluate ROPF, since this requires knowledge of the secret key. This is a relatively serious requirement, which is rarely satisfied in practice. Further, their definition also requires uniformly random distribution of the pre-image, which they call the "uniformity assumption," and the authors warn that their analysis may not hold in cases where this assumption does not hold. Further they note (and we fully agree) that this assumption usually will *not* hold in practice.

ROPF (and Boldyreva's OPE) are shown to be one-way in the restricted sense described above. However, standard stronger security notions do not provably hold for ROPF. In this work, we discuss these notions and the impact on system security, using the example of webmail.

Models of Secure Function Evaluation

In this work, we consider the security of reactive systems (i.e., systems which engage in computation in several rounds, adjusting their input for each round). While we do not prove, or even formally specify, the precise security properties of secure webmail or other applications, we believe that it is beneficial to present a high-level discussion of the relevant cryptographic security models, along with their justification.

The Semi-Honest Model: Intuition and Justification

The adversaries we consider are very close to *semi-honest* (sometimes also called *passive*). Intuitively, a semi-honest adversary follows the protocol specification exactly, yet attempts to gain additional information by analyzing "everything he sees" during protocol execution, i.e., the input received, randomness, and the transcripts of messages received. Although the semi-honest adversary is far weaker than a *malicious* or *active* one (which may arbitrarily deviate from the protocol specification), uses of and research into the semi-honest model is well justified. The semi-honest model assumes all players are semi-honest.

First, protection against adversaries that are only semi-honest is often sufficient for real world applications. Indeed, it is often the case that there is a certain sufficient mutual trust among the players executing a protocol. At the same time, it is hard to ensure that all traces of computation are destroyed after completion of the protocol, even if both parties wish to do so. This is because of the complex structure of the networks, virtual memory, and caching mechanisms. Information is almost always stored in several places. A trace of a secure execution in the semi-honest model will be of limited help to an adversary who might later hack into a player's computer and obtain the information. The hacking will, of course, compromise the private information of the player who was hacked, but the private information of the other players will remain hidden because of the security properties of the (completed) protocols. This justification scenario is valid for cloud services, where the cloud provider is well trusted (but of course cannot be considered to be invulnerable to malicious attacks).

Further, it is often the case that reputation is of high importance to businesses and even to private parties. In many scenarios, the payoff for actively cheating is not very high, while the cost of being caught is significantly higher (e.g., destroyed reputation or the prospect of legal action). Even though the probability of being caught (e.g., by a random inspection of software or other methods) may be small, this alone may be a sufficient deterrent to active cheating. Semi-honest cheating, however, is often impossible to detect, and thus protection is needed against semi-honest players. This justification scenario is also valid for less-trusted service providers.

Finally, protocol behavior may effectively be hidden in a large software or hardware system (there exist heuristic methods for obfuscating an execution process), and the cost of amending the behavior of such systems may be higher than the potential benefit. This justification scenario may apply to cases where the client himself creates a hardware or software device and places it in the hands of the server for operation.

While the above discussion justifies certain direct uses of protocols that run securely in the semi-honest model, research into this model can also serve as an important stepping-stone to achieving fully secure protocols in the malicious model. There are tools

(bit commitments, zero-knowledge proofs, and basic protocols which allow players to securely choose random bits) which allow *automatic* compilation of a protocol secure in the semi-honest model into a protocol computing the same functionality securely in the malicious model. Intuitively, players first commit to their input and securely choose their random tapes. Then, for every message sent by the semi-honest protocol, a zero-knowledge proof is added, convincing other players that the message is formed properly (i.e., consistently with the protocol, player's randomness, and previous messages). With a negligible probability of proving a false statement, parties are forced to follow the protocol, and thus their cheating capabilities lie in the semi-honest model. With respect to practical applications—our main topic of interest—this automatic compilation greatly increases overhead, usually to the point of making the resulting protocols completely impractical. However, we note that heuristic improvements, or protection only against certain malicious behaviors, may often be added at a reasonable cost, constituting a good tradeoff. In fact, the approaches we advocate for webmail follow this line of algorithm design.

Further, different players may have different levels of trust. For example, in an auction system, a bidder may trust an established auction house to act semi-honestly. At the same time, neither he nor the auction house might have such trust in other bidders, thus requiring protection against malicious bidders. In such cases, it is often most efficient to design semi-honest protocols, and then selectively add efficient protection against certain malicious behaviors of certain players.

The Malicious Model

The malicious model is probably the most natural model that guarantees no cheating is possible (or, rather, that cheaters will be caught with overwhelming probability). While attractive at first glance, the malicious model is very restrictive, and requires very significant overhead, as compared to a plain semi-honest model. For example, in the malicious model, Yao's garbled circuit solution has overhead approaching a multiplicative factor of several hundred compared to semi-honest implementations.

Our Hybrid Model

Relatively recently, a model which is a compromise between the semi-honest and malicious settings has been introduced by Aumann and Lindell [2]. This model, which is called *covert*, allows cheating as long as there is a constant fixed probability of the cheater being caught. This relatively small security reduction allows efficient protocols, with costs within a small multiplicative factor of that of the semi-honest model (when certain generic secure computation techniques, amenable to the covert modification, such as the garbled circuit, are used.)

In this work, we consider basic algorithms based on OPE, and we don't see a natural opportunity to adjust our algorithms to fit the covert model at low cost. Further, we don't consider the malicious model in its full generality (or with strict formalism). Instead, we consider several natural, low-risk, high-payoff malicious behaviors, and discuss techniques to mitigate them.

The Webmail Application

As noted above, OPE helps enable a variety of applications where security concerns may prevent sharing the data in plaintext. However, since we aim to demonstrate and analyze in this work, even the use of an ideal OPE implementation will not necessarily imply a secure application. We chose the following webmail scenario to illustrate the use and advantages of OPE, as well as its limitations.

The primary appeal of order-preserving encryption is the fact that it offers the ability to encrypt data in a way that allows searches to be performed without possession of the secret key. Additionally, and in contrast to deterministic-encryption alternatives, the queries supported by OPE include not only equality searches (searching for a specific keyword) but also range queries, which are critical for a variety of applications. OPE can be viewed as a tool somewhat similar to fully-homomorphic encryption, in that it can repeatedly operate on encrypted data. It is weaker than FHE since the manipulation primitive is limited to equality checking and comparisons. However, even more importantly, in contrast with FHE, the OPE program evaluator knows the result of the comparisons, which leaks certain information. This information,

when aggregated over the life of the system, and especially when combined with information that may be available externally, may reduce or even completely dismantle security provided by OPE. However, OPE offers truly practical efficiency, and is in fact one of the very few available scalable crypto-computing tools.

We consider the following scenario: a webmail application provided as a service by a mail server *M*. We will assume that the mail server may be interested in datamining client emails, and even may be interested in attacks against an individual client. However, *M* cannot afford to be caught cheating. Therefore, we will consider *M* to be a semi-honest adversary, who may be able to perform certain "safe" deviations from the protocol, which are very unlikely to get him caught.

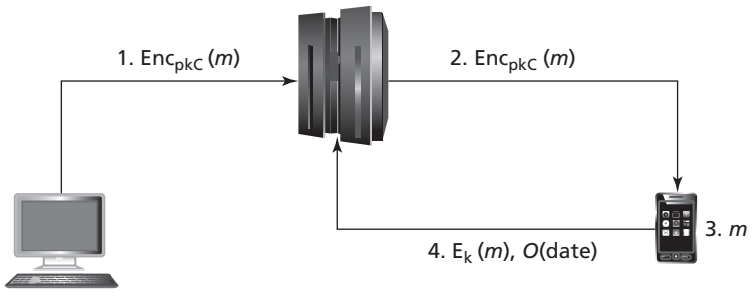
The first issue to note is that of the architecture: the mail server *M* processes and delivers all of *C*'s electronic mail, and hence may have seen all the incoming and outgoing mail. In that case, the protections we are instituting may be moot. We note that this need not be an issue: email may be sent and received in encrypted form in the first place. In this case, email processing would need to involve *C* to re-encrypt with OPE, as described next.

The Webmail Architecture

When a sender *S* wishes to send an email to a client *C* through a mail server *M*, he encrypts the message in an end-to-end fashion, so that *M* cannot access its content. The most natural scenario here relies on the use of public key infrastructure (PKI), as follows. *C* owns a public/private key pair pk_R/sk_R, which is published through the PKI. *S* encrypts the mail *m* with *C*'s public key pk_R (e.g., using hybrid encryption Advanced Encryption Standard (AES) with RSA-Optimal Asymmetric Encryption Padding (OAEP)) resulting in $Enc_{pk_R}(m)$. Upon receiving $Enc_{pk_y}(m)$, the mail server *M* forwards it to *C* (of course, *M* cannot infer anything about *m*, or modify it). With its private key sk_R, *C* decrypts the message locally and retrieves *m*.

The main benefit of web-based email is its universal access. To leverage the ubiquity of the cloud, *C* stores *m*, encrypted with *C*'s symmetric key, at *M*'s server. The encryption algorithm here can simply be AES. However, the use of OPE is required in order for *C* to take advantage of *M*'s essential services, such as mail sorting and classification. To reconcile security with functionality, *C* uses OPE to encrypt certain additional fields which are appended to the message, including the ones used to perform searches and range queries, such as the date. Now, at least in the toy scenario described above, we have succeeded: *M* cannot access *C*'s messages but can reply to range queries on the fields encrypted with OPE.

This scheme is summarized in **Figure 1**.



In this scenario, the message *m* is encrypted by the sender *S* under the public key of the receiver *C*, then it is sent through the mail server *M* to *C*. *C* decrypts the message, processes it, and sends it back to the server with a different encryption with the addition of some fields (e.g., the date) encrypted using an order preserving encryption denoted by *O*.

Figure 1.
The webmail scenario.

Attacking Webmail Application via OPE Queries

We will now show that most of the security for the natural OPE-based webmail architecture described above falls apart in real environment. Let's assume that M has access to certain side-channel information about the OPE ciphertexts stored on his servers. We will discuss how he may obtain this information in the next section.

Known- and Chosen-Plaintext Attacks

Known-plaintext attacks (KPA) and chosen-plaintext attacks (CPA) help form the basis of standard notions on the security of cryptosystems.

Informally, a KPA occurs when the adversary obtains samples, via the so-called *KPA oracle*, of both the plaintext and its encrypted version (but the adversary has no control over which pairs he obtains). An encryption scheme is said to be KPA-secure if the adversary cannot gain any information about the plaintexts of the ciphertexts, which were never returned by the KPA oracle. We note that KPA security is a relatively weak notion, and deterministic encryption schemes, such as AES, satisfy this notion. Recall, the weakness of deterministic encryption is in the fact that two encryptions of the same plaintext are equal (if we encrypt a document word-by-word to enable exact-keyword searching, the attacker can identify many of the words by frequency analysis).

The chosen plaintext attack is similar to the KPA, except that the adversary can now choose plaintexts and obtain their encryptions by calling a CPA-encryption oracle. An encryption scheme is said to be CPA-secure if the adversary cannot gain any information about the plaintexts of any ciphertexts, regardless of prior interaction with the CPA oracle. We note that CPA security is a standard notion in cryptography, which is satisfied by *probabilistic* encryption schemes.

Unfortunately, OPE does not (and cannot, because of the order-preserving functionality it must provide) satisfy either of the above notions.

KPA and CPA Attacks on OPE

In this section we describe how KPA and CPA oracles can help break the security of OPE, and hence we observe that OPE is not secure against these attacks. In the next section, we will show that these oracles are in fact present in natural webmail implementations.

With OPE, the knowledge of a single plaintext/ciphertext pair splits both the domain and the range of the OPE function in two. (Once the OPE key is fixed, we can think of OPE_k as a fixed function). Suppose the OPE domain is $[1, m]$ and that the adversary knows the encryption $OPE_k(x)$ of a plaintext x (where $1 \leq x \leq m$). Then, upon seeing a ciphertext y, the adversary can compare y with $OPE_k(x)$, and determine whether the plaintext corresponding to y lies in $[1, x]$ or in $[x, m]$. This split can be viewed as akin to replacing the OPE on the existing domain $[1, m]$ and implementing two OPE schemes, one on domain $[1, x]$ and the other on domain $[x, m]$. As with any deterministic scheme, the useable security of an OPE scheme with a smaller domain is weaker than that of a scheme on a larger domain (in our example, $[1, m]$). Clearly, the more plaintext/ciphertext pairs the adversary knows, the more it can subdivide the domain, and the weaker the security offered by OPE becomes.

Going further, and assuming that the adversary can perform arbitrary adaptive queries to the encryption oracle (i.e., the CPA attack), he can decrypt any ciphertext y with only $\log(m)$ oracle queries, using a divide-and-conquer approach. Further, the domain/range subdivisions obtained by the adversary persist throughout the lifetime of the encryption key, which allows decryption with even fewer than $\log(m)$ oracle queries. In some circumstances, such as in a small domain (whether *a priori* or due to domain subdivision), and with the availability of other auxiliary information, decryption may be possible even without additional oracle queries.

We stress that these attacks apply to *any* implementation of OPE, including the "ideal OPE" described by Boldyreva et al.

For completeness, we also must mention chosen-ciphertext attacks (CCA). A CCA attack is similar to a CPA attack, except that the adversary is *additionally* given access to a decryption oracle. CCA-security is the strongest notion. Informally, it guarantees that an adversary cannot manipulate ciphertexts in a way that it would decrypt to a value related to the original plaintext. OPE is not secure against CCA attacks, and, certainly, having access to the CCA decryption oracle helps the attacker. These attacks should be kept in mind when designing systems; however, CCA decryption

oracles are slightly less natural in the webmail setting, as we will argue in the next section.

KPA and CPA Oracles and Other Auxiliary Information in Webmail

The previous section clearly illustrates the security threat brought forward by simple attacks, given access to KPA or CPA oracles. In this section we show that it is in fact quite hard (and, in some scenarios, impossible) to block access to these oracles.

We first observe that it is in fact rarely the case that the adversary has absolutely no information about the plaintext. Indeed, the fields in webmail suitable for OPE encryption include the date/time (represented as a 32-bit integer, for example), and sender's name or domain. An adversarial M can certainly estimate (and in the simple architecture described above, do so with reasonable confidence) the plaintexts corresponding to the OPE ciphertexts received. This knowledge effectively corresponds to the KPA attack.

Further, M can perform the following malicious attack. He can pretend to be a legitimate sender, and send an arbitrary message to C containing the plaintext x that M wants encrypted with OPE. This is easily achieved simply by encrypting x with the public key pk_R of the receiver C, delivering this encryption as part of received mail, and receiving from C $OPE_k(x)$, according to the protocol. This attack is relatively low-risk, since it is hard for C to distinguish (especially in an automated manner, and especially if the email is carefully crafted) such CPA oracle emails from, say, a "regular" unsolicited email. At the same time, this is a high-payoff attack, since it may allow nearly arbitrary access to the CPA-encrypt oracle.

The CCA decryption oracle, or at least, some restricted form of it, can also be accessed by M, for example, as follows. The application run by the client C (which is in possession of the secret key) automatically responds to encrypted mail delivered to it. It decrypts the messages and displays them for the user. Either the application or the user may take certain actions, visible to M based on the content of the decrypted email fields. For example, the application may automatically notify the server of a decryption failure or a sorting discrepancy, and the user may respond to an email. This information, collected by M, may help him infer the plaintext contents of the OPE-encrypted data it sent, which is similar to the CCA decryption oracle.

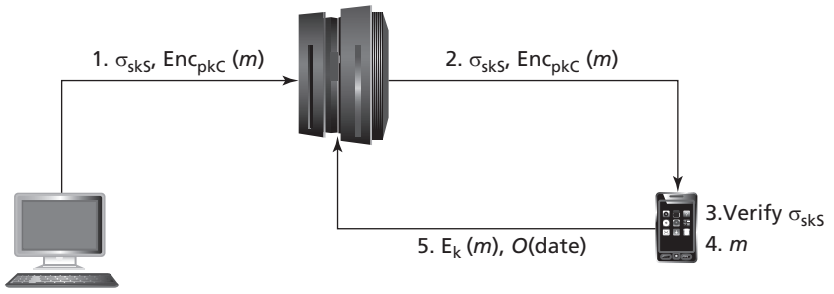
With respect to general auxiliary information, the malicious mail server M may have other side-channel information, which may help it reduce the entropy of (its view of) the client's data. For example, any personal, group, or statistical information available about a particular client helps predict his communication patterns. This includes dates and times, the vocabulary used in emails, and names within his circle of correspondents.

Our Approach

As discussed earlier in this paper, it is impossible, or, at best, extremely difficult, to design a complex usable system which relies on OPE and achieves provable security. As argued and presented in detailed examples in the previous section, the auxiliary information obtained by the mail server M simply by serving the client C, may be easily used to break the security of OPE employed by C. We have also shown that this vulnerability is inherent with the use of OPE.

In cryptographic analysis, it is customary to "give" the adversary any and all information with respect to inputs from the honest players that the adversary requests. A system or protocol is considered to be secure if its execution does not reveal anything to an adversary that he didn't already know (or is allowed to learn during the execution). On the other hand, as soon as a non-negligible amount of information is leaked, the system is considered fully insecure. This is a very powerful approach, and its strength is in fact required in order to build secure systems from secure blocks. The downside of this all-or-nothing approach to security is that in situations and systems such as webmail, cryptography cannot be used for formal analysis due to the auxiliary (or side-channel) information that may be available to the malicious players. Further, it is very difficult even to estimate the amount of leaked information.

Therefore, our approach and recommendation is to aim to design systems which reduce the amount of usable auxiliary information and achieve heuristic security with only rough estimates of the level of security achieved. In the current state-of-the-art in cryptography, this is the best we can hope for.



In addition to encrypting the message *m*, the sender *S* signs it with its private key sk_S. The signature σ_{skS} is verified by the receiver *C* who only processes mail from trusted sources. In particular, *C* will not process mail originating from *M*.

Figure 2.
First approach: signed mail.

Next, we will present several simple ideas that would greatly reduce the amount of available side-channel information, and make acquiring it less easy and more risky with respect to exposure.

Restricting Access to OPE Encryption Oracle

One of the most powerful threats to OPE webmail security that we identified is the adversary's ability to gain access to the OPE encryption oracle. This access stems from the fact that *M* can generate messages of his choice and ask *C* to encrypt them with OPE. We propose several ways to limit, and even largely eliminate this access.

Signed mail. One natural and powerful approach we advocate is for the client to only encrypt mail coming from trusted sources with OPE. This is most easily achieved by public key based signature authentication of each message, as illustrated in **Figure 2**. If public key infrastructure (PKI) is available, this approach should be easy to implement. However, even without PKI, public keys of known senders can be stored by *C* each time a contact is added or the first message is received from a particular sender. Man-in-the-middle attacks by *M* are, of course, possible in this scenario, but they are relatively hard to perform, and carry a high risk of detection by *S* or *C*.

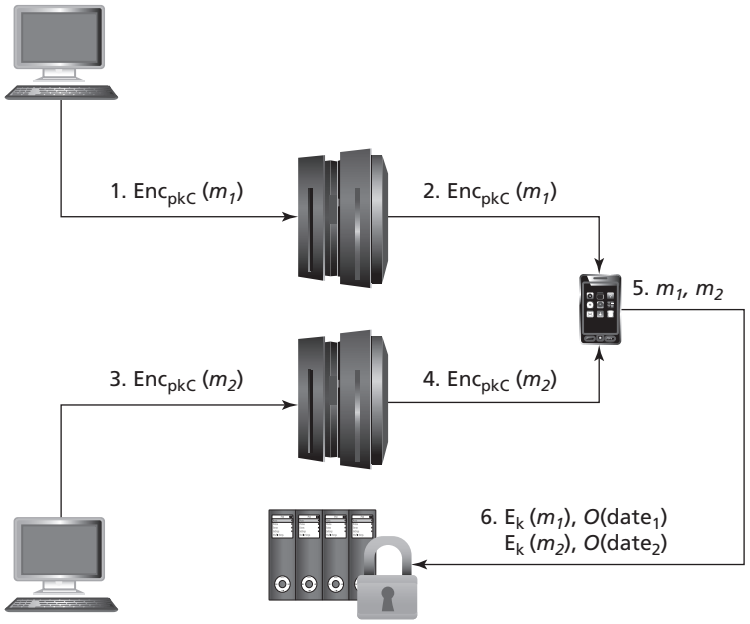
A natural way for *M* to circumvent this protection is to become a trusted sender or to collude with one. We note that in the settings where email privacy is most desired, such as with corporate email, this may be hard to achieve. More importantly, this is a malicious action, and one that is relatively easy to detect, which is a strong deterrent against this attack.

Separation of roles. Another approach to restricting access to the OPE encryption oracle is based on the observation that the mail service plays two different roles:

- Acting as a relay for mail, i.e., transferring mail that is sent or received to its legitimate recipient, and
- Serving as a display interface and a storage mechanism for mail.

In our scenario, the mail relay sees the encrypted mail with plaintext headers and other available information, such as size and timestamps. At the same time, the mail storage mechanism sees OPE-encrypted fields. It is the combination of these two roles that allows *M* to associate plaintexts and ciphertexts, and results in access to the OPE encryption oracle.

Therefore, we suggest separating these two roles and instituting a requirement that two non-colluding entities to perform them. The first role is the core functionality required from a mail server and it has to be performed by *M*, while the second role could be performed by any storage service with appropriate user interface (UI) support. We argue that another mail server, *M* ', is a natural candidate for mail storage. Indeed, the advantage of mail services over simple storage systems is that they not only offer ubiquitous access to stored mail, but they also have a webmail-specific front end with a UI that supports sorting by



Here we assume that the receiver C receives mail from two accounts, processed respectively by mail servers M_1 and M_2. C processes the received messages and then stores them on a different repository server denoted by M_R.

Figure 3.
Second approach: separation of roles.

fields, searching, and importing mail archives (such as .pst archives). Further, since OPE encryptions can be viewed, or at least encoded, as "regular" plaintext fields such as timestamps or sender names, any changes necessary to support OPE encryption by existing webmail servers may be very minimal.

In summary, we envision using two different mail accounts from different providers:

- The first mail account, which we call the *reception account*, is used to relay encrypted mail.
- The second mail account, which we call the *repository account*, is used only as a mail storage and UI interface, and will not directly receive or send any mail. (All direct mail received for this account should be disregarded).

We further note that a single repository account can be naturally used to support multiple reception accounts, as shown in **Figure 3**. As shown, the client C has several reception accounts (with the same or different providers). C fetches the mail from the different accounts, processes them, and then stores them all on the same repository account (offered by a separate provider). The use of the repository account thus also may be positioned as a desirable feature, as it enables efficient searches across separate accounts.

To mask the timestamp and mail size information from M, C may choose to upload mail in batches, and pad the email fields as needed.

In summary, separating roles as advocated in this section is a powerful tool for greatly limiting the side-channel information and OPE encryption and decryption queries available to M.

Probabilistic OPE (POPE)

Deterministic encryption is traditionally viewed as necessarily insecure since the revolutionary work of Goldwasser and Micali on probabilistic encryption in 1982. However, due to its functional requirements, OPE is defined to be deterministic. We propose that in some settings it is possible to introduce randomization to OPE, and trade some of its functionality for increased security.

Indeed, in a deterministic scheme, every encryption of a plaintext value x would be mapped to the same ciphertext y. Once the adversary decrypts y, all encryptions of x are uncovered. Not so when randomness is used for encryption: each probabilistic OPE encryption of x maps to a different y_r, and decryption of a particular y_r does not allow full confidence in decrypting all encryptions of x.

The tradeoff is that now, while

$$\forall x_1, x_2 \in D_1, x_1 < x_2 \Rightarrow POPE(x_1) < POPE(x_2),$$

the converse preserves the order in the less strict sense:

$$\forall x_1, x_2 \in D_1, POPE(x_1) \leq POPE(x_2) \Rightarrow x_1 < x_2.$$

With respect to POPE constructions, we propose two simple variants for adding randomness to OPE.

The first idea is to artificially extend the size of the plaintext domain by first mapping (with order preservation) elements from the original domain to a larger domain, and then applying OPE from the larger domain. The domain extension can be done, e.g., by appending n random bits to the bit representation of the elements in the original domain. It is easy to see that this extension is order-preserving, and that it enjoys the POPE benefit described above.

The second approach is to build on a deterministic OPE, as follows. We set $POPE(x)$ to be equal to a randomly-chosen element from interval $[OPE(x), \ldots, OPE(x+1)]$. Decryption in this case will generally be more costly, and require the use of several calls to OPE in a divide-and-conquer manner.

Conclusion

Order-preserving encryption is often seen as a powerful cryptographic tool that can be securely plugged into existing systems. We demonstrated, with clear and detailed examples, that this most often is not the case. We provided a high-level architecture of a webmail system that relies on OPE to protect client data from the web server. We identified several of the most serious sources of insecurity, and demonstrated how to largely mitigate their effects.

This work can also be seen as a high-level overview of the area of secure computation, its application to practice, and of the privacy issues that arise when designing real life systems.

References

[1] R. Agrawal, J. Kiernan, R. Srikant, and Y. Xu, "Order Preserving Encryption for Numeric Data," Proc. ACM SIGMOD Internat. Conf. on Management of Data (SIGMOD '04) (Paris, Fra., 2004), pp. 563–574.

[2] Y. Aumann and Y. Lindell, "Security Against Covert Adversaries: Efficient Protocols for Realistic Adversaries," Proc. 4th Theory of Cryptography Conf. (TCC '07) (Amsterdam, Nld., 2007) LNCS vol. 4392, pp. 137–156.

[3] A. Boldyreva, N. Chenette, Y. Lee, and A. O'Neill, "Order-Preserving Symmetric Encryption," Proc. 28th Internat. Conf. on Advances in Cryptology (EUROCRYPT '09) (Cologne, Ger., 2009), LNCS vol. 5479, pp. 224–241.

[4] A. Boldyreva, N. Chenette, and A. O'Neill, "Order-Preserving Encryption Revisited: Improved Security Analysis and Alternative Solutions," Proc. 31st Conf. on Advances in Cryptology (CRYPTO '11) (Santa Barbara, CA, 2011), LNCS vol. 6841, pp. 578–595.

[5] Y. Ding and K. Klein, "Model-Driven Application-Level Encryption for the Privacy of E-Health Data," Proc. 5th Internat. Conf. on Availability, Reliability, and Security (ARES '10) (Krakow, Pol., 2010), pp. 341–346.

[6] C. Gentry, "Fully Homomorphic Encryption Using Ideal Lattices," Proc. 41st ACM Symp. on Theory of Comput. (STOC '09) (Bethesda, MD, 2009), pp. 169–178.

[7] V. Kolesnikov, "Advances and Impact of Secure Function Evaluation," Bell Labs Tech. J., 14:3 (2009), 187–192.

[8] H. Liu, H. Wang, and Y. Chen, "Ensuring Data Storage Security Against Frequency-Based Attacks in Wireless Networks," Proc. 6th IEEE Internat. Conf. on Distrib. Comput. in Sensor Syst. (DCOSS '10) (Santa Barbara, CA, 2010), LNCS vol. 6131, pp. 201–215.

[9] W. Lu, A. L. Varna, and M. Wu, "Security Analysis for Privacy Preserving Search of Multimedia," Proc. 17th IEEE Internat. Conf. on Image Processing (ICIP '10) (Hong Kong, Hkg., 2010), pp. 2093–2096.

[10] R. A. Popa, C. M. S. Redfield, N. Zeldovich, and H. Balakrishnan, "CryptDB: Protecting Confidentiality with Encrypted Query Processing," Proc. 23rd ACM Symp. on Operating Syst. Principles (SOSP '11) (Cascais, Prt., 2011), pp. 85–100.

[11] Q. Tang, "Privacy Preserving Mapping Schemes Supporting Comparison," Proc. ACM Cloud Comput. Security Workshop (CCSW '10) (Chicago, IL, 2010), pp. 53–58.
[12] C. Wang, N. Cao, J. Li, K. Ren, and W. Lou, "Secure Ranked Keyword Search over Encrypted Cloud Data," Proc. 30th IEEE Internat. Conf. on Distrib. Comput. Syst. (ICDCS '10) (Genova, Ita., 2010), pp. 253–262.
[13] A. C.-C. Yao, "How to Generate and Exchange Secrets," Proc. 27th Annual Symp. on Foundations of Comput. Sci. (FOCS '86) (Toronto, Ont., Can., 1986), pp. 162–167.

(Manuscript approved June 2012)

VLADIMIR KOLESNIKOV is a member of technical staff in Bell Labs' Enabling Computing Technologies research domain in Murray Hill, New Jersey. He received his Ph.D. in computer science from the University of Toronto. His research interests include secure multiparty computation, key exchange, the foundations of cryptography, and network security. His work is connected to the practice of cryptography. He has worked on securing channels in smart grid and Worldwide Interoperability for Microwave Access (WiMAX) networks, biometric authentication, secure and private databases, and other subjects. Dr. Kolesnikov has published his work in top cryptographic and security conferences and journals. He has served on the program committees for several international cryptography conferences. He is an editor of the WiMAX "Server Certificate Profile" and "Device Certificate Profile" standards documents.

ABDULLATIF SHIKFA is the scientific advisor and deputy head of the security research department at Bell Labs, in Nozay, France. Before joining Alcatel-Lucent, he was a research engineer with the EURECOM Network and Security team working on the EU FP6 project Haggle, where he designed original security protocols for challenging environments such as mobile opportunistic networks and radio frequency identification (RFID). His research interests and experience span a wide range of topics in information and communication security from trust and cooperation enforcement, to secure routing, through protection of user privacy mainly by applying advanced cryptographic primitives to enable computation on encrypted data. Dr. Shikfa obtained his Ph.D. from Telecom ParisTech. He is also an alumnus of Ecole Polytechnique, of the University of Nice at Sophia Antipolis, and of ENST-EURECOM where he earned three separate M.Sc. degrees in three consecutive years. His research results led to over 10 publications in international scientific conferences and journals, and to several invited talks. Dr. Shikfa is also a member of the IEEE and ACM, and has served as technical program committee member for major security and communication conferences. ◆