Chapter 7.12
Provable Security for Outsourcing Database Operations

Sergei Evdokimov
Humboldt-Universität zu Berlin, Germany

Matthias Fischmann
Humboldt-Universität zu Berlin, Germany

Oliver Günther
Humboldt-Universität zu Berlin, Germany

ABSTRACT

Database outsourcing has become popular in recent years, although it introduces substantial security and privacy risks. In many applications, users may not want to reveal their data even to a generally trusted database service provider. Several researchers have proposed encryption schemes, such as privacy homomorphisms, that allow service providers to process confidential data sets without learning too much about them. In this paper, the authors discuss serious flaws of these solutions. The authors then present a new definition of security for homomorphic database encryption schemes that avoids these flaws and show that it is difficult to build a privacy homomorphism that complies with this definition. As a practical compromise, the authors present a relaxed variant of the security definition and discuss arising security implications. They present a new method to construct encryption schemes for exact selects and prove that the resulting schemes satisfy this notion.

DOI: 10.4018/978-1-61350-323-2.ch7.12

INTRODUCTION

The alternative [to rigorous proofs of security] is to design systems in an ad hoc manner, and to simply hope for the best. As we have seen, this approach often ends in disaster, or at least, an embarrassing mess.-- Victor Shoup, IBM, 1998

In this paper, we assess cryptographic solutions to the problem that some client party (Alex) wants to outsource database operations on sensitive data sets to a service provider (Eve) without having to trust her. Forming contracts and relying on law enforcement are options, but for various reasons their effectiveness is limited and the costs for negotiations, auditing and prosecution are considerable (Boyens & Günther, 2002). Alex would prefer to encrypt his data in a way that enables Eve to perform operations on the ciphertext yielding encrypted results, which Alex could in turn decrypt. All this should ideally take place without revealing anything to Eve about the plaintext data or the operations.

Consider two examples:

1. If the service provider (say, Peoplesoft) changes owners, it is unclear whether the new owner (say, Oracle) is still legally bound by the same contract, regardless of pre-existing privacy policies.[1] As Amazon has phrased it in a similar context:[2]

> In the unlikely event that Amazon.com Inc., or substantially all of its assets are acquired, customer information will of course be one of the transferred assets.

Alex has good reasons to be worried about this situation if he has to give away sensitive data in the clear. However, if the data were never exposed to any service provider, no contract would be required in the first place.

2. Celebrity Paris Hilton's personal phone book was stolen and leaked to the public in 2005.[3]

> The data had been stored in a T-Mobile server farm and used from a mobile device. One theory of what happened is that hackers got access to the central server, where the data was stored in plaintext. If a secure privacy homomorphism had been used, the hackers would have obtained only useless ciphertext.

The security requirements of Alex heavily depend on the application. The strongest notion of confidentiality keeps every single bit of information on queries as well as data secret from Eve, no matter how many queries she can observe, or how clearly she breaks the contract. This notion would allow for doing business with arbitrarily malicious service providers, but we will see shortly that it is impossible to achieve.

Confidentiality against an adversary that can do computations on the ciphertext (even with encrypted output) is difficult. Traditionally, confidentiality is defined precisely in terms of the adversary's incapability of doing any such computations. Even if the adversary cannot necessarily understand the outcome of the computation, confidentiality and processability are strong antagonists.

Worse, the application to databases is extreme in this respect: Relational algebra is a rich formalism that imposes a complex structure on the data being processed. If Eve were supposed to process arbitrary terms of relational algebra, she would need to have a lot of structural information on the ciphertext, which, together with a plausible amount of context knowledge, would most likely allow her to deduce at least fractions of the secret data.

The idea that encryption schemes for situations like these could exist has been brought up almost 30 years ago (Rivest, Adlerman & Dertouzos, 1978) under the name *privacy homomorphism*. Our aim is to find privacy homomoprhisms for database outsourcing that transform relational data sets and queries into ciphertext such that (*i*) the data is *securely hidden* from Eve, although she has unlimited access to the ciphertext; and (*ii*) Eve can compute ciphertext results from ci-