New Security Models and Provably-Secure Schemes for Basic Query Support in Outsourced Databases

Georgios Amanatidis[*] Alexandra Boldyreva[*] Adam O'Neill[*]

Abstract

In this paper, we take a closer look at the security of outsourced databases (aka Database-as-the-Service or DAS), a topic of emerging importance. DAS allows users to store sensitive data on a remote, untrusted server and retrieve desired parts of it on request. At first we focus on basic, exact-match query functionality, and then extend our treatment to prefix-matching and, to a more limited extent, range queries as well. We propose several searchable encryption schemes that are not only practical enough for use in DAS in terms of query-processing efficiency but also provably-provide privacy and authenticity of data under new definitions of security that we introduce. The schemes are easy to implement and are based on standard cryptographic primitives such as block ciphers, symmetric encryption schemes, and message authentication codes. As we are some of the first to apply the provable-security framework of modern cryptography to this context, we believe our work will help to properly analyze future schemes and facilitate further research on the subject in general.

Keywords: database security, searchable encryption, symmetric authenticated encryption.

1 Introduction

MOTIVATION. Outsourcing data to off-site database service providers is becoming an attractive, cost-effective option for many organizations, the main reasons for the trend being outlined in a recent business whitepaper [41]. In this setting (also known as Database-as-a-Service or DAS), a client stores data on a remote untrusted database server and queries the server in order to receive required portions of the data. Usually this data is stored in the form of a relational database, each divided into records (or tuples) with attributes (or fields). The basic system requirements are (1) query support, (2) computation and communication efficiency for both client and server, and (3) data security. Note that the latter requirement is particularly important in DAS, as data often contains sensitive financial, medical, or intellectual information and the server cannot be trusted. Indeed, ensuring security in DAS is an important research topic that has been receiving increasing attention [26, 38, 28, 27, 21, 22, 38, 3, 29, 2, 30, 32, 8]. Security may even be required by law (see HIPAA rules [1]).

The problem is that these requirements are in conflict with each other. For example, consider encrypting the data with a secure encryption scheme that hides *all* information and is always randomized (i.e. same messages yields completely different ciphertexts). This does not allow the user to even form a query about any set of records smaller than the the whole database. Indeed, it turns out that even addressing just the basic exact-match (point) queries is a non-trivial task if one wants to treat security in a systematic, not ad-hoc, way.

PREVIOUS WORK. Searching on encrypted data has been a topic of multiple relevant works in the cryptographic community, which focus mainly on exact-match queries but in an unsatisfactory way for our context. In particular, the schemes of [42, 23, 25, 15, 18, 20] provide strong security guarantees (typically revealing only the user access pattern) while allowing a server to answer exact-match queries, but doing so, unlike for the schemes developed in the database community, *requires the server to scan the whole database for each query*, yielding unacceptably-slow performance for medium-size to very large databases. The schemes of [20] get around this problem by requiring the (paying) client to know all keywords and all data beforehand and pre-computing a static index for

[*]Georgia Institute of Technology, USA, amana@math.gatech.edu, {aboldyre,amoneill}@cc.gatech.edu

the server that does not allow to treat relational databases. A fundamental question thus becomes what is the best guaranteed security that can be achieved without compromising general efficiency and functionality. The work of [8] recently raised this problem in the asymmetric (public-key) setting, where users explicitly consist of "senders" and "receivers," and provided new security definitions and provably-secure solutions for exact-match queries. We consider this problem entirely in the more-natural symmetric-key setting where a client (which may be a large group of users, e.g. in a business) both stores and queries its own data on an untrusted server.

Research on this subject done in the database community focuses on the first two requirements and provides encryption schemes with attractive functionality, namely efficient and optimized indexing and flexible query support e.g. for numerical range, comparison, or aggregation queries [38, 3, 26, 22, 28, 29, 27, 32]. In contrast, the security of these schemes is far less clear. Many utilize cryptographic primitives such as deterministic encryption, order-preserving hash functions and encryption schemes, which have not been studied by cryptographers, and without scrutinizing their security. For example, using a deterministic encryption scheme for point queries sounds like a reasonable idea, because then forming a point query is feasible and the server can efficiently index and locate the ciphertexts. But what scheme should be used? One common suggestion (see e.g. [29, 2]) is to use DES or AES. But these are block ciphers for short plaintexts of at most 128 bits. But if a database field stores a larger data, say a barcode information, then it is not clear how to encrypt longer ones. It would be natural to apply the block cipher block-by-block, but then the adversary will see when the underlying plaintexts have common blocks, which is an unnecessary leak of information. Similarly, fixing the randomness in an arbitrary encryption mode (e.g. CBC) will leak more information than needed.

A noteworthy exception in this body of work is a recent paper by M. Kantarcioglu and C. Clifton [30], which calls for a new direction of research that aims for "efficient encrypted database and query processing with *provable* security properties." This involves making new cryptographic definitions to model precisely what we want our schemes to do and how much security we expect them to achieve. The methodology allows one to *guarantee* that a given scheme will remain secure according to a clear definition of security against *all* possible attacks captured by the definition that can be realized within reasonable time (many years) under some widely-accepted computational assumptions. The work of [30] provides a first step in this direction. As they observe, unless one lowers the security bar from the previous cryptographic solutions a linear scan of the database on each query is fundamentally necessary. But the above discussion suggests we must be careful to not go too far. On the other hand, the security definition proposed in [30] requires the use of server-side trusted, tamper-resistant hardware to achieve. This may limit practicability and also undermines one of the points of using provable security, namely to guarantee that if an adversary violates the definition of security they must have solved a difficult computational problem. (They might instead have found a hole in the trusted hardware.)

OVERVIEW OF CONTRIBUTIONS. In a broad sense, our goal is to narrow the gap between query-processing-efficient but ad-hoc schemes with unclear security and schemes with strong security guarantees but with unsuitable functionality. We review the provable-security methodology in Section 2. Then to start with, we consider exact-match queries (i.e. with boolean conditions involving only equalities). In Section 4, we formulate what algorithms and properties constitute an *efficiently-searchable authenticated encryption or ESAE* scheme that will allow a server to process such queries, when used to separately encrypt each searchable field, with, unlike for previous cryptographic-community schemes, query-processing efficiency comparable to that for unencrypted databases.

As opposed to previous works in the database community, we go significantly beyond explaining why some attacks do or do not work in order to develop a rigorous and general *foundation* for our understanding of security. Observe that while typically encryption hides all partial information about the data (which is still true for previous searchable schemes in the cryptographic community, and homomorphic encryption schemes in a basic model of security), ESAE cannot because some information needs to be leaked to allow efficient query processing. Hence we formulate a new definition of security that captures the intuition that no adversary should be able to learn *any* useful information about the data within reasonable time, beyond what is unavoidable for the given functionality, namely when two ciphertexts correpond to equal plaintexts; we argue that permitting false-positive results cannot help to hide this correlation in practice. Our definition moreover captures a notion of authenticity that ensures attributes values are not modified or added over the network or at the server side without the user noticing.[1] Thus in a sense we provide the strongest possible notion of security one can reasonably ask for without relying on trusted hardware as in [30]. Note that we do not explicitly model security in the terms of a client-database interaction but always instead simply derive security in this context from that of the "ideal" cryptographic object in question. (This step is crucially absent in [32].) In Section 5 we propose and analyze two exact-match ESAE constructions meeting our definition.

Then in Section 6 we extend our framework to treat prefix-matching and, to a more limited extent, range queries as well, for which we formulate an important open problem. We propose a novel security definition for the former that ensures, analogous to the case of exact-match queries, that no information about the data is leaked except for what is unavoidable. We then present a new prefix-preserving ESAE scheme and prove its security under standard assumptions. We show how our scheme can be used for range queries as well and generalize and refine the approach of [32], pinpointing exactly what level of security can be guaranteed by such schemes in the context of DAS. Finally, we conclude with some possibilities for extending our schemes to additionally process aggregate queries.

2 The Provable-Security Methodology

We start with an overview of the provable-security framework. Cryptographic protocols were often designed by trial-and-error, where a scheme is implemented and used until some flaws found and fixed, if possible, and the revised scheme is used until new flaws are found, and so forth. A revolutionary and superior "provable-security" approach was originally proposed by Goldwasser and Micali [24]. The approach requires a formal definition of a security goal (e.g., data privacy) for a given cryptographic object (e.g., an encryption scheme). A security definition comprises a formal description of adversarial capabilities (what an adversary knows and can do) and of what an adversary must do to break the scheme, designed to capture the real-world intuitive security goal in an "imaginary" formal setting. For example, a definition of a semantically-secure symmetric encryption scheme captures the intuition that no partial information is leaked from the ciphertexts and formally requires that no efficient adversary should be able to distinguish between encryptions of two messages, even if the adversary can choose these two messages and request to see ciphertexts of other different messages of its choice.

A proof of security then shows by reduction that a given scheme satisfies the definition under widely accepted assumptions (e.g., that factoring big composite numbers is hard). That is, the proof rigorously *transforms* an adversary against the scheme into a corresponding one against the underlying hard problem. The proof thus shows that *the only way* to break the scheme is by breaking the underlying assumption about the hard problem. In other words, a provably-secure scheme *guarantees* to withstand *all infinitely many* possible attacks implicitly captured by the security definition as long as the underlying hard problem remains hard.[2] (This is rather like showing that a given computational problem is likely to remain intractable for a long time because one can transform any instance of, say, SAT, to a corresponding instance of this problem) Symmetric encryption schemes, e.g. ciphertext-block-chaining (CBC) or counter (CTR) modes, are usually built from block ciphers, e.g. DES or AES, and are proven secure assuming the block cipher is a pseudorandom function (PRF). A comprehensive overview of the provable-security framework can be found in [5].

3 Preliminaries

NOTATION. We refer to members of $\{0,1\}^*$ as strings. If X is a string then $|X|$ denotes its length in bits and if

[1]The issues of authenticity for the database and the records as a whole, and ensuring that the server returns all the current, requested data, are outside our scope and can be dealt with the methods of [33, 36, 37, 31]. Typically the server may get to learn and sell the data, but it gains little from returning the wrong data to its paying clients, and one may assume the server is protected from other adversaries. In any case, the schemes we propose will achieve our notion of integrity without any additional computational or communication cost.

[2]Of course, a practical cryptographic scheme can always be broken by exhaustive key search, so one should only consider efficient adversaries with reasonable computational power and running time, say several decades on a powerful machine.

X, Y are strings then $X \| Y$ denotes the concatenation of X and Y. If S is a set then $X \xleftarrow{\$} S$ denotes that X is selected uniformly at random from S. For convinience, for any $k \in N$ we write $X_1, X_2, \ldots, X_k \xleftarrow{\$} S$ as shorthand for $X_1 \xleftarrow{\$} S, X_2 \xleftarrow{\$} S, \ldots, X_n \xleftarrow{\$} S$. If A is a randomized algorithm then $A(x, y, \ldots; R)$, or $A(x, y, \ldots)$ for short, denotes the result of running A on inputs $x, y, \ldots$ and with coins R, and $a \xleftarrow{\$} A(x, y, \ldots)$ means that we choose R at random and let $a = A(x, y, \ldots; R)$. Oracle access, when given to algorithms (and denoted by superscript), is done as a "black-box," meaning the algorithms see only the input slots provided to it, but neither the code for the oracle algorithm nor other, fixed inputs. A family of functions is a map $F\colon \{0,1\}^b \times \{0,1\}^c \to \{0,1\}^c$, where we regard $\{0,1\}^b$ as the *keyspace* for the function family in that a *key* $k \in \{0,1\}^b$ induces a particular function from this family, which we denote by $F(k, \cdot)$.

SYMMETRIC ENCRYPTION AND MESSAGE AUTHENTICATION.

Definition 3.1 [Symmetric encryption scheme] *A symmetric encryption scheme $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ with associated message space $\mathsf{MsgSp}(\mathcal{SE})$ consists of three algorithms. (1) The randomized key generation algorithm $\mathcal{K}$ returns a secret key sk; we write $sk \xleftarrow{\$} \mathcal{K}$. (2) The (possibly randomized) encryption algorithm $\mathcal{E}$ takes input the secret key sk and a plaintext m to return a ciphertext; we write $C \xleftarrow{\$} \mathcal{E}(sk, m)$ or $C \leftarrow \mathcal{E}(sk, m; R)$. If $C = \mathcal{E}(sk, m, R)$ for some coins R then we say C is a* valid *ciphertext for m under sk. (3) The deterministic decryption algorithm $\mathcal{D}$ takes the secret key sk and a ciphertext C to return the corresponding plaintext or a special symbol $\perp$ indicating that the ciphertext was invalid; we write $m \leftarrow \mathcal{D}(sk, C)$ (or $\perp \leftarrow \mathcal{D}(sk, C)$.)*

Consistency: we require that $\mathcal{D}(sk, (\mathcal{E}(sk, m)) = m$ for all $m \in \mathsf{MsgSp}(\mathcal{SE})$.

The idea behind security of encryption is that an adversary against a scheme should not be able to deduce anything about the underlying message (except its length, which encryption cannot hide), upon seeing the ciphertext, even if it has some *a priori* information of its choice about the message. This intuition is captured via a notion of "indistinguishability" of encryptions [10] that requires that no efficient adversary should be able to distinguish between encryptions of two messages, even if the adversary can choose these two messages and request to see ciphertexts of other different messages of its choice.

Definition 3.2 [Security of encryption] *Let $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a symmetric encryption scheme with $\mathsf{MsgSp}(\mathcal{SE})$. Let $\mathcal{LR}$ (left-or-right) be the "selector" that on input m_0, m_1, b returns m_b. The scheme $\mathcal{SE}$ is said to be* secure against chosen-plaintext attack *or* ind-cpa *if for every efficient adversary B the value called the advantage of B $\mathbf{Adv}^{\text{ind-cpa}}_{\mathcal{SE},B}$ is sufficiently small, where*

$$\mathbf{Adv}^{\text{ind-cpa}}_{\mathcal{SE},B} \quad = \quad \Pr[\,\mathbf{Exp}^{\text{ind-cpa-0}}_{\mathcal{SE},B} = 0\,] - \Pr[\,\mathbf{Exp}^{\text{ind-cpa-1}}_{\mathcal{SE},B} = 0\,]$$

and the experiments above are defined for $b \in \{0,1\}$ and an ind-cpa adversary B who is required to query messages of equal length and in $\mathsf{MsgSp}(\mathcal{SE})$, as follows:

```
Experiment $\mathbf{Exp}^{\text{ind-cpa-}b}_{\mathcal{SE},B}$
    $sk \xleftarrow{\$} \mathcal{K}$ ; $d \xleftarrow{\$} B^{\mathcal{E}(sk, \mathcal{LR}(\cdot,\cdot,b))}$ ; Return $d$   ▮
```

Typically, one also gives a "chosen-ciphertext" version of the definition where B has access to a special kind of decryption oracle, and it is accepted that encryption should defend against such attacks. It turns out that we will not require this from standard encryptions schemes for our constructions to achieve this type of security. Note that the definition does not forbid B from using the same message as both components in a query to its left-or-right encryption oracle, thus in this way it can also obtain the encryption of a message of its choice. Note that no deterministic encryption scheme is ind-cpa.

Also note that we purposely do not mathematically define an "efficient" adversary and how "small" the advantage should be. This will vary according to the particular appliation. For example, guaranteeing that all adversaries whose running time is up to 2^{60} in some fixed RAM model of computation have maximum advantage 2^{-20} would usually be considered sufficient.

Definition 3.3 [MAC] *A deterministic message authentication code or MAC scheme $\mathcal{MAC} = (\mathcal{K}, \mathcal{M}, \mathcal{V})$ with associated message space $\mathsf{MsgSp}(\mathcal{MAC})$ consists of three algorithms. (1) The randomized key generation algorithm $\mathcal{K}$ returns a a secret key sk; we write $sk \xleftarrow{\$} \mathcal{K}$. (2) The deterministic mac algorithm $\mathcal{M}$ takes input the secret key sk and a plaintext m to return a "mac" for m; we write $\sigma \leftarrow \mathcal{M}(sk, m)$.*[3] *(3) The deterministic verification algorithm $\mathcal{V}$ takes the secret key sk, a message m, and a mac σ to return a bit $b \in \{0,1\}$. We write $b \leftarrow \mathcal{V}(sk, m, \sigma)$. In the case that the above b is 1 we say that σ is a valid mac for m under sk.*

Consistency: we require that $\mathcal{V}(sk, m, (\mathcal{M}(sk, m)) = 1$ for all $m \in \mathsf{MsgSp}(\mathcal{SE})$.

More generally one can permit $\mathcal{M}$ to flip coins as well, but most practical message authentication schemes (e.g., CMAC or HMAC) are deterministic which is important for our context. Thus in this paper "MAC" means "deterministic MAC."

The standard definition of security of MACs, unforgeability under chosen-message attacks (or uf-cma) requires that no efficient adversary that sees macs of the messages of its choice can produce a valid mac for a new message.

Definition 3.4 [Security of MACs] *A MAC scheme $\mathcal{MAC} = (\mathcal{K}, \mathcal{M}, \mathcal{V})$ is said to be unforgeable against chosen-message attack or uf-cma if for every efficient adversary B the value $\mathbf{Adv}^{\text{uf-cma}}_{\mathcal{MAC},B}$ called* advantage *of B is sufficiently small, where*

$$\mathbf{Adv}^{\text{uf-cma}}_{\mathcal{MAC},B} \;=\; \Pr[\,\mathbf{Exp}^{\text{uf-cma}}_{\mathcal{MAC},B} = 1\,] \text{ and the experiment is defined as}$$

$$\begin{array}{l}\textbf{\textit{Experiment}}\ \mathbf{Exp}^{\text{uf-cma}}_{\mathcal{MAC},B} \\ \quad sk \xleftarrow{\$} \mathcal{K}\,;\ (m,\sigma) \xleftarrow{\$} B^{\mathcal{M}(sk,\cdot),\mathcal{V}(sk,\cdot,\cdot)}\,;\ \textit{Return } \mathcal{V}(sk,m,\sigma)\end{array}$$

and B is not allowed to query m to its mac oracle. ▌

We will also use an additional property of MACs, namely privacy preservation, originating recently in [11], that requires the outputs of the MAC to hide information about the messages similarly to encryption.

Definition 3.5 [Privacy-preserving message authentication] *[6, 11] A MAC scheme $\mathcal{MAC} = (\mathcal{K}, \mathcal{M}, \mathcal{V})$ is said to be* privacy-preserving *if for every efficient adversary B the value called the advantage of B $\mathbf{Adv}^{\text{pp-mac}}_{\mathcal{MAC},B}$ is sufficiently small, where*

$$\mathbf{Adv}^{\text{pp-mac}}_{\mathcal{MAC},B} \;=\; \Pr[\,\mathbf{Exp}^{\text{pp-mac-0}}_{\mathcal{MAC},B} = 0\,] - \Pr[\,\mathbf{Exp}^{\text{pp-mac-1}}_{\mathcal{MAC},B} = 0\,]$$

and the experiments above are defined for the adversary B and , $b \in \{0,1\}$ as follows

$$\begin{array}{l}\textbf{\textit{Experiment}}\ \mathbf{Exp}^{\text{pp-mac-}b}_{\mathcal{MAC},B} \\ \quad sk \xleftarrow{\$} \mathcal{K}\,;\ d \xleftarrow{\$} B^{\mathcal{M}(sk,\mathcal{LR}(\cdot,\cdot,b))}\,;\ \textit{Return } d\end{array}$$

Above $\mathcal{LR}$ is the oracle that on input m_0, m_1, b returns m_b; and we require that for any sequence of oracle queries $(m_{1,1}, m_{1,2}), \ldots, (m_{q,1}, m_{q,2})$ that B can make to its oracle, there does not exist any $m_{i,1} = m_{j,1}$ or $m_{i,2} = m_{j,2}$ for $i \neq j$ and moreover $|m_{i,1}| = |m_{i,2}|$ for all i. ▌

For our schemes, it will be useful to consider MACs that are uf-cma and also privacy preserving, and most practical MACs are.

[3]In the cryptographic literature what we are denoting by $\mathcal{M}$ is often denoted by and called the *tagging* algorithm, but we will want to use "tag" for something else.

4 Efficiently-Searchable Authenticated Encryption

WHAT IS ESAE. We now define the syntax of an ESAE (Efficiently-Searchable Authenticated Encryption) scheme, which can be used to encrypt the attribute values of a database and allow efficient processing of exact-match queries on encrypted data.

Definition 4.1 [ESAE] *Let $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a symmetric encryption scheme. We say that $\mathcal{ESAE} = (\mathcal{K}, \mathcal{E}, \mathcal{D}, \mathcal{F}, \mathcal{G})$ an* authenticated efficiently-searchable encryption *(*ESAE*) scheme if $\mathcal{K}, \mathcal{E}, \mathcal{D}$ are the algorithms of a regular encryption scheme and $\mathcal{F}, \mathcal{G}$, are deterministic efficient algorithms where the former takes a secret key and message as input and the latter takes a ciphertext and:*
(1) Completeness:

$$\Pr\left[\, sk \xleftarrow{\$} \mathcal{K} \,;\, f_1 \leftarrow \mathcal{F}(sk, m_1) \,;\, g_1 \leftarrow \mathcal{G}(\mathcal{E}(sk, m_1)) \;:\; f_1 = g_1 \,\right] = 1 \text{ and}$$

(2) Soundness:

$$\Pr\left[\, sk \xleftarrow{\$} \mathcal{K} \,;\, (m_0, m_1) \xleftarrow{\$} \mathcal{M}_{\mathcal{SE}} \;:\; \mathcal{F}(sk, m_0) = \mathcal{G}(\mathcal{E}(sk, m_1)) \,\right] \text{ is sufficiently small },$$

for every message $m_1 \in \mathsf{MsgSp}(\mathcal{SE})$ and every efficient randomized algorithm $\mathcal{M}_{\mathcal{SE}}$ that outputs distinct messages $m_0, m_1 \in \mathsf{MsgSp}(\mathcal{SE})$. We refer to the output of $\mathcal{F}, \mathcal{G}$ as the tag *of a message m or a corresponding ciphertext C.*

At a high level, the algorithm $\mathcal{F}$ is used by the user to form queries, and $\mathcal{G}$ is needed by the server to be able to index the encrypted data *a priori*, using the standard data structures, and locate records on request (see below), for which it is crucial that $\mathcal{F}, \mathcal{G}$ are not randomized. Thus the completeness property ensures that encrypted data can be efficiently searched, and the soundness property ensures that false positives do not occur too often so that post-processing is efficient.

USAGE. We focus on the case that the soundness probability in the definition so small that each ciphertext essentially has a unique tag; we will address increasing the number false-positive results later. Observe that in this case completeness implies that the server will be able locate ciphertexts (or tags) of attribute values that correspond to same ones used to form the query so that query processing can be done server-side in logarithmic-time in the database size, meaning this time has not gone up over unencrypted data, and that this moreover can be done using essentially the same interface and dynamic index structures for the tuples as for unencrypted relational databases, which is appealing to implementors. Updates can thus be handled similarly, and the schema and metadata can also be encrypted in this way.

Such exact-match functionality can also be used to build various other useful more-complicated query types. These include equijoin and group-by, the latter of which is especially useful for example in supporting multi-faceted search that projects among various dimensions (e.g. features/types of products). Moreover, the server can *ipso facto* compute counts over the data, which would also be useful in this context for example to support a product search interface that shows there are, say, 100 CRT and 200 LCD monitors in the database, and 100 15”, 100 17”, and 100 20” monitors. You click on LCD monitors link and it now shows 50 15”, 75 17”, and 75 20” such monitors.

SECURITY OF ESAE. Efficient “searchability” (ensured by the completeness property) necessarily violates the standard ind-cpa security for encryption. Thus we aim to provide a relaxed definition suitable for given functionality. Let us first stay with the simpler case where no false-positive results are allowed (e.g. when users connect over a low bandwidth channel or when the server needs to compute some aggregate query over a certain set of attribute values). As in this case completeness implies that the server (and the adversary) will always be able to see what ciphertexts correspond to equal plaintexts, and a security definition should ensure that this is *all* the adversary can learn. To this end we design an indistinguishability experiment (cf. Definition 3.2) where we disallow the adversary from seeing ciphertexts of equal messages such that it can trivially succeed. The adversary can also mount chosen-ciphertext attacks according to a relaxed chosen-ciphertext-security definition [4, 16] that is suitable for our application. For integrity of the data, we also want to require that no efficient adversary can produce a new ciphertext or change the existing one without the user noticing, which corresponds to a notion of ciphertext-integrity for authenticated encryption [12].

Definition 4.2 [Security of authenticated efficiently-searchable encryption] *Let $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D}, \mathcal{F}, \mathcal{G})$ be an ESAE scheme. Let $\mathcal{LR}$ (left-or-right) be the selector that on input m_0, m_1, b returns m_b. Let B be an adversary who is given access to two oracles (called lr-encryption and the decryption oracles). For $b \in \{0, 1\}$ define the experiment:*

```
Experiment $\mathbf{Exp}^{\text{ind-esae-}b}_{\mathcal{SE},B}$
    $sk \xleftarrow{\$} \mathcal{K}$ ; $d \xleftarrow{\$} B^{\mathcal{E}(sk,\mathcal{LR}(\cdot,\cdot,b)),\mathcal{D}(sk,\cdot)}$
    If $m \neq \perp$ was returned from $\mathcal{D}(sk,\cdot)$ at any point then $d \leftarrow b$
    Return $d$
```

We call B an esae adversary *if for any sequence of queries $(m_{1,1}, m_{1,2}), \ldots, (m_{q,1}, m_{q,2})$ that B can make to its lr-encryption oracle, there does not exist any $m_{i,1} = m_{j,1}$ or $m_{k,2} = m_{l,2}$ for $i \neq j, k \neq l$ such that $m_{i,2} \neq m_{j,2}$ or $m_{k,1} \neq m_{l,1}$, in addition to the usual requirements that $|m_{i,1}| = |m_{i,2}|$ for all i and if B does not query the decryption oracle on a ciphertext that has the same tag as any ciphertext that has been returned by the lr-encryption oracle. The* advantage *of an esae adversary B is defined as follows:*

$$\mathbf{Adv}^{\text{ind-esae}}_{\mathcal{SE},B} \quad = \quad \Pr[\,\mathbf{Exp}^{\text{ind-esae-}0}_{\mathcal{SE},B} = 0\,] - \Pr[\,\mathbf{Exp}^{\text{ind-esae-}1}_{\mathcal{SE},B} = 0\,].$$

The ESAE scheme $\mathcal{SE}$ is said to be esae-secure *if for every efficient privacy adversary B the function $\mathbf{Adv}^{\text{ind-esae}}_{\mathcal{SE},B}$ is sufficiently small.* ▌

We note the similarity of ESAE to deterministic authenticated encryption (DAE), studied in [40] in the context of transporting (encrypted) symmetric keys. However, the definition of security for DAE in [40] is shown there be equivalent to that for "pseudorandom injections," and we will see that an ESAE scheme need not be pseudorandom nor deterministic (an injection), see Construction 5.1 and remark.

DISCUSSION. In the context of DAS, the server receives queries with tags for the data, the former of which it would have computed itself, thus the definition of security we provide essentially guarantees that the server cannot learn anything about the data of the user beyond its occurrence profile (or distribution), i.e. how many times a given attribute value (without knowing anything else about it) occurs in the database and in which records, even if it is one of only two possible such values that it can pick itself, and analogously the user access pattern. This implicitly assumes however that the adversary cannot mount a chosen-plaintext attack *after* seeing the database or otherwise obtain *a priori* information about the data other than the message space (however our strong definition ensures security would still hold if unrelated formerly-secret data that used to be stored in the database was later published), which would be highly undesirable as it would allow to correlate all the places a plaintext occurs as well as semantic correlation with other attribute values. We consider this to be a reasonable tradeoff for the functionality and efficiency our schemes achieve: in our setting, such an attack would be difficult for the server (in particular one can treat the server and network as separate adversaries, adding an "outer" layer of ind-cca encryption just for client-server communication) and can moreover be addressed through other security measures.

As for authenticity (aka. integrity) of ciphertexts, our definition guarantees integrity in that any modification or substitution (malicious or not) to the encrypted data is detected by the user. We not that authenticity is ensured at the field level, and not on the record level or for the entire database; an adversary can still, for example, switch (encrypted) attribute values stored in different records. If the data is updated and returned as whole records, then one can simply authenticate at the record level instead. In many applications, though, the server can be trusted to return the correct ciphertexts to its paying customers (even when it may try to learn and sell their data). Thus one should mainly protect against non-adversarial transmission or storage errors, and our definition does it.

INCREASED FALSE-POSITIVES. Next let us consider if it is possible to hide the occurrence profile of the data and still achieve comparable query-processing efficiency. Indeed, it seems intuitive that permitting false positive results (i.e. relaxing the soundness condition in Definition 4.1) via a "bucketization" technique where a fixed number of randomly-chosen plaintexts correspond to each tag, [35, 34, 17], though requiring the client to do more work to filter out these false-positives, would allow a proportional increase security by preventing the adversary from correlating equal plaintexts. But we claim that this intuition is not always correct; in practice such information may still be leaked. To see this, consider the *a posteriori* probability of a plaintext occurring a certain number of times given an occurrence distribution on the buckets; the "farther" the latter is from the uniform distribution means a better estimate on the plaintext occurrence profile, and one cannot expect anything close to the uniform distribution in practice. One solution would be make the bucket distribution instead depend on that of the input, but in particular as noted in [35] this would require impractical communication cost between client and server as this distrbution changes over time, and it is noted in [32] that such mappings are typically not efficiently computable, making storing and managing them impractical. It is also natural to assume that most frequently occurring attribute values are accessed most often, so analogously this kind of bucketization would not hide the user access pattern from the server in a meaningful way, either. Moreover, the typical case that one would expect a uniform distribution on the occurrence numbers of each attribute value for a field is when the attribute value is unique to a record, e.g. employee ID number, but in this case such identifiers need to be keyed for use in updates or joins and so could not be bucketized in any case.

COMPARISON TO KANTARCIOGLU-CLIFTON. The security definition of [30] guarantees that an adversary (e.g. the server) cannot distinguish between two queries whose results sets have the same size, whereas ours reveals which records are accessed by such queries. This hold even with respect to extremely powerful adversaries who can mount chosen-ciphertext attacks, whereas our definition applies to somewhat more passive adversaries, which we nevertheless believe is reasonable for the given application. On the other hand, the definition of [30] requires server-side trusted hardware to achieve.

5 Proposed Constructions and Their Security Analyses

Note that the straight-forward schemes we discussed in the Introduction are insecure under Definition 4.2. Accordingly we propose new schemes and analyze their security.

MAC-AND-ENCRYPT. We first present an "off-the-shelf" way to construct an ESAE scheme from any encryption and MAC schemes and then analyze its security and comment on implementation.

Definition 5.1 [Mac-and-encrypt construction] *Let $\mathcal{SE} = (\mathcal{K}_E, \mathcal{E}, \mathcal{D})$ be a (standard) symmetric encryption scheme and $\mathcal{MAC} = (\mathcal{K}_M, \mathcal{M}, \mathcal{V})$ be a message authentication code. Then we define a new symmetric encryption scheme $\mathcal{SE}^* = (\mathcal{K}^*, \mathcal{E}^*, \mathcal{D}^*, \mathcal{F}, \mathcal{G})$, whose constituent algorithms work as follows:*

- *$\mathcal{K}^*$ sets $sk_M \xleftarrow{\$} \mathcal{K}_M$ and $sk_E \xleftarrow{\$} \mathcal{K}_E$, then outputs $sk_M \| sk_E$.*
- *$\mathcal{E}^*$ on input $sk_M \| sk_E, m$, sets $\sigma \leftarrow \mathcal{M}(sk_M, m)$ and $C \xleftarrow{\$} \mathcal{E}(sk_E, m)$, then outputs $\sigma \| C$.*
- *$\mathcal{D}^*$ on input $sk_M \| sk_E, \sigma \| C$, first sets $m \leftarrow \mathcal{D}(sk_E, C)$ and then $b \leftarrow \mathcal{V}(sk_M, m, \sigma)$. It outputs m if $b = 1$ and $\perp$ otherwise.*
- *$\mathcal{F}$ and $\mathcal{G}$ on inputs $sk_M \| sk_E, m$ and $\sigma \| C$, respectively, return $\mathcal{M}(sk_M, m)$ and σ.*

We first argue that $\mathcal{SE}^*$ is an ESAE scheme if $\mathcal{MAC}$ is uf-cma. We let $\mathcal{F}$ and $\mathcal{G}$ from the definition be the algorithms that on inputs $sk_M \| sk_E, m$ and $\sigma \| C$, respectively, return $\mathcal{M}(sk_M, m)$ and σ. Clearly this satisfies completeness. The soundness condition relies on the uf-cma security of $\mathcal{MAC}$. Namely, suppose $\mathcal{MAC}$ is uf-cma but there is an algorithm $\mathcal{M}_{\mathcal{SE}}$ that outputs m_0, m_1 such that $\mathcal{M}(sk_M, m_0) = \mathcal{M}(sk_M, m_1)$ with high probability. This violates uf-cma security as follows. We construct a uf-cma adversary B as per Definition 3.4 that first runs $\mathcal{M}_{\mathcal{SE}}$ to receive its output (m_0, m_1) then queries its signing oracle for $\mathcal{M}(sk, m_0)$ to get back σ, and finally itself returns (m_1, σ). By the forgoing assumption on $\mathcal{M}_{\mathcal{SE}}$ this adversary has high uf-cma advantage, a contradiction.

Theorem 5.2 *Let $\mathcal{SE} = (\mathcal{K}_E, \mathcal{E}, \mathcal{D})$ be a symmetric encryption scheme and $\mathcal{MAC} = (\mathcal{K}_M, \mathcal{M}, \mathcal{V})$ be a deterministic MAC. Then let $\mathcal{SE}^* = (\mathcal{K}^*, \mathcal{E}^*, \mathcal{D}^*, \mathcal{F}, \mathcal{G})$ be the mac-and-encrypt ESAE scheme defined according to Definition 5.1. We have that $\mathcal{SE}^*$ is esae-secure if $\mathcal{SE}$ is ind-cpa and $\mathcal{MAC}$ is uf-cma and privacy-preserving. More precisely, let A be an esae adversary against $\mathcal{SE}^*$ making at most q_e lr-encryption queries and at most q_d decryption queries. Then there exists a mac-privacy adversary B_1, a uf-cma adversary B_2 against $\mathcal{MAC}$ and ind-cpa adversary B_3 against $\mathcal{SE}$ such that*

$$\mathbf{Adv}^{\text{ind-esae}}_{\mathcal{SE}^*,A} \quad \le \quad 2\mathbf{Adv}^{\text{pp-mac}}_{\mathcal{MAC},B_1} + 2q_d\mathbf{Adv}^{\text{uf-cma}}_{\mathcal{MAC},B_2} + \mathbf{Adv}^{\text{ind-cpa}}_{\mathcal{SE},B_3}. \tag{1}$$

The proof is given in Appendix A.

There are many efficient and standardized provably-secure symmetric encryption and MAC schemes that can be used to build an ESAE scheme according to Definition 5.1. Our recommendations for encryption schemes include CBC and CTR (aka the counter or XOR) encryption modes based on the AES block cipher, which are proven to be ind-cpa under the assumption that AES is a pseudorandom function (PRF) [10]. For MACs, one can use SHA-1 or SHA-256 and AES-based HMAC or CMAC (a variation of CBC-MAC), proven uf-cma assuming the underlying hash function is collision-resistant or PRF and the block cipher is PRF [9, 6, 14]. Theorem 5.2 implies that the resulting mac-and-encrypt ESAE is secure under the respective assumptions.

We remark that in database literature (e.g. [26]), some proposed solutions for this problem suggest to use a "random one-to-one mapping" whose output is included with a ciphertext, in order to facilitate "searchability." Thus one interesting implication of the above result is that such a map need not be random, or even pseudorandom, in order to achieve the best-possible notion of security. This may not be merely of theoretical interest, as recent results [6] have established that security of popular HMAC instatiations (which in particular is a PRF) now relies on somewhat non-standard assumptions that may not in fact hold [19].

ENCRYPT-WITH-MAC. We now present a construction that is more computation-efficient on the client side and more communication-efficient over the network. This can be crucial, for example, when users have a low-bandwidth connection to the database or are connecting via a battery-constrained device [36]. The idea is to use the mac of the plaintext "inside" the encryption, namely as the randomness used in the encryption algorithm of a standard encryption scheme.

Definition 5.3 [Encrypt-with-mac construction] *Let $\mathcal{SE} = (\mathcal{K}_E, \mathcal{E}, \mathcal{D})$ be a (standard) symmetric encryption scheme and $\mathcal{MAC} = (\mathcal{K}_M, \mathcal{M}, \mathcal{V})$ be a deterministic MAC. Then we define a new symmetric encryption scheme $\mathcal{SE}^* = (\mathcal{K}^*, \mathcal{E}^*, \mathcal{D}^*, \mathcal{F}, \mathcal{G})$, whose constituent algorithms work as follows:*

- *$\mathcal{K}^*$ sets $sk_M \xleftarrow{\$} \mathcal{K}_M$ and $sk_E \xleftarrow{\$} \mathcal{K}_E$, then outputs $sk_M \| sk_E$.*
- *$\mathcal{E}^*$ on input $sk_M \| sk_E, m$, sets $\sigma \leftarrow \mathcal{M}(sk_M, m)$ and $C \leftarrow \mathcal{E}(sk_E, m; \sigma)$, then outputs C.*
- *$\mathcal{D}^*$ on input $sk_M \| sk_E, C$, first sets $m \leftarrow \mathcal{D}(sk_E, C)$. It outputs m if $C = \mathcal{E}(sk_E, m; \mathcal{M}(sk_M, m))$ and $\perp$ otherwise.*
- *$\mathcal{F}$ is same as $\mathcal{E}^*$. $\mathcal{G}$ on input C returns C.*

To see that $\mathcal{SE}^*$ is an ESAE scheme, we note that the completeness requirement is clearly satisfied and the probability in the soundness requirement is zero here due to the consistency requirement in Definition 3.1.

Ideally, we would like to prove that the above construction is ESAE-secure assuming that $\mathcal{MAC}$ is a uf-cma and $\mathcal{SE}^*$ is ind-cpa secure. However, slightly stronger assumptions turns out to be needed, but they are met by practical schemes anyway. First, we will need the mac algorithm of $\mathcal{MAC}$ to be a pseudorandom function (PRF). Naturally, this requires a mac to "look like random bits" without the secret key, a well-studied notion formalized as follows.

Definition 5.4 *A family of functions is a map $F\colon \{0,1\}^b \times \{0,1\}^c \to \{0,1\}^c$, where we regard $\{0,1\}^b$ as the* keyspace *for the function family in that a* key $k \in \{0,1\}^b$ *induces a particular function from this family, which we denote by $F(k,\cdot)$. The family F is said to be* pseudorandom *(or a PRF) if for every efficient adversary B given oracle access to a function, its advantage*

$$\mathbf{Adv}^{\mathrm{prf}}_{F,B} = Pr\left[B^{F(k,\cdot)} = 0\right] - \Pr\left[B^{Q(\cdot)} = 0\right]$$

is sufficiently small, where $F(k,\cdot)$ is the oracle for a random instance of F (specified by a randomly chosen key k) and $Q(\cdot)$ is the oracle for a truly random function with the domain and range of $F(k,\cdot)$. Pseudorandom permutations (PRPs) are defined analogously, and in this case the adversary B above is also given an inversion oracle.

Most known MACs are PRFs.

To define the assumption needed for encryption, let us say that an encryption scheme $\mathcal{SE} = (\mathcal{K},\mathcal{E},\mathcal{D})$ has a *max-collision probability* $mc_{\mathcal{SE}}$ if we have that:

$$\Pr\left[\mathcal{E}(sk,m,R_1) = \mathcal{E}(sk,m,R_2)\right] \leq mc_{\mathcal{SE}}\ ,$$

for every $m \in \mathsf{MsgSp}(\mathcal{SE})$, where the probability is taken over the random choices of the key sk and coins R_1, R_2 (chosen independently).

All practical encryption schemes satisfy the above property. The proof of the following is in Appendix A.

Theorem 5.5 *Let $\mathcal{SE} = (\mathcal{K}_E,\mathcal{E},\mathcal{D})$ be a (standard) symmetric encryption scheme and $\mathcal{MAC} = (\mathcal{K}_M,\mathcal{M},\mathcal{V})$ be a deterministic MAC. Let $\mathcal{SE}^* = (\mathcal{K}^*,\mathcal{E}^*,\mathcal{D}^*)$ be the encrypt-with-mac ESAE scheme defined via Definition 5.3. Then $\mathcal{SE}^*$ is esae-secure if $\mathcal{MAC}$ is a PRF and $\mathcal{SE}$ is ind-cpa and has sufficiently small max-collision probability. More precisely, let A be an esae adversary against $\mathcal{SE}$ making at most q_e encryption queries and q_d decryption queries. Then there exists an ind-cpa adversary B against $\mathcal{SE}$ and a prf adversary D against $\mathcal{MAC}$ such that:*

$$\mathbf{Adv}^{\mathrm{ind\text{-}esae}}_{\mathcal{SE}^*,A} \quad \leq \quad \mathbf{Adv}^{\mathrm{ind\text{-}cpa}}_{\mathcal{SE},B} + 2\mathbf{Adv}^{\mathrm{prf}}_{\mathcal{MAC},D} + \frac{2q_d}{mc_{\mathcal{SE}}}\ .$$

The same recommendations for the underlying schemes (CBC, CTR modes, and HMAC and CMAC) we gave for the mac-and-encrypt construct apply here. As we mentioned, CBC and CTR are proven to be ind-cpa assuming the base block cipher is PRF. Randomized CBC and CTR have max-collision probability 2^{-128} when used with AES and the counter-based CTR has zero max-collision probability. HMAC was recently proved to be a PRF assuming the underlying hash function is PRF [6], and CMAC is known to be PRF assuming the base block cipher is PRF; Theorem 5.5 implies that the resulting encrypt-with-mac ESAE scheme is secure under these respective assumptions.

We remark that our construction is similar to the SIV ("synthetic initialization vector") construction for deterministic authenticated encryption (DAE) in [40]. Indeed, it is straightforward to check that a secure DAE scheme as defined in [40] is also secure as an ESAE scheme. However, our construction and analysis is in fact somewhat more general than the SIV construction, which pertains only to some "initialization-vector-based" symmetric encryption schemes (including CBC and CTR) that implicitly guarantee to meet the max-collision requirement that we pinpoint for security.

6 Prefix-Preserving Efficiently-Searchable Authenticated Encryption

PREFIX-MATCHING QUERIES. We extend our ESAE framework to encryption that allows to efficiently process prefix-matching queries, i.e. locating records whose attribute value starts with a given prefix, for example all phone numbers starting with country-code 86. (Observe that processing such queries [as well as range queries, which we address later] still takes linear time in the database size if the data is encrypted with an ESAE developed in the previous section.) While arguably not as fundamental a query type, they can be very useful in some contexts and we are able to pinpoint the "right" ideal object for supporting such queries, which generalizes an approach previously suggested for supporting range queries [32] and will also help us develop an understanding of the basic security challenges for the latter.

Our treatment builds on the study of "online ciphers" (so-called because they can be used on streaming data without buffering) in [7], which we view as deterministic length-preserving encryption schemes whose input is composed of fixed-length blocks (which we view as "characters" in the prefixes), where the ith block of the output depends only on the first i blocks of the input. Thus if two plaintexts agree on their first k characters then so do their ciphertexts. Following Definition 4, to show this implies efficient prefix-searchability (via appropriate server-side index structures for the tuples) we can make functions $\mathcal{F}, \mathcal{G}$ return the encryption of an l-character prefix and the first l characters of a ciphertext; completeness one and soundness zero follows from the fact that encryption is deterministic.

In our construction the characters of a prefix are of the input-length for an underlying block cipher (e.g. 64 bits or 4 UTF-16 characters using DES-variants). At the cost of revealing more information to the server for more flexible granularity of prefixes in the queries a *bitwise* prefix-preserving scheme of Xu et al. [43] can similarly be used here (an issue we will return to later), which makes one block cipher computation per *bit* of the input. We observe that this may be too inefficient for, say, text files as input. Moreover, as for our previous schemes our construction also achieves ciphertext-integrity, whereas it seems hard to somehow modify the former to achieve such a notion.[4]

SECURITY. The stronger security definition for an online cipher in [7] requires it to be indistinguishable from an "ideal" object that is a function drawn at random from a family of all possible such "online" permutations with the corresponding domain, even when given access to the corresponding "inverter" decryption oracle. Note that for example applying encryption character-by-character is completely insecure: encryptions of "HAT" and "BAT" should look totally unrelated in this setting despite sharing a suffix. We also formulate an additional property of ciphertext-integrity, and thus the encryption algorithm should contain some redundancy at the end so the ciphertext is verifiable. For our definition, we use an ideal object that encrypts a message with a random block appended, and the decryption oracle in the ideal experiment always returns $\perp$ to capture the intuition that the adversary should not be able to create a new valid ciphertext. The novelty of our definition is its generality: it uses only the ideal object in question and without any specific redundancy.

Definition 6.1 [Security of prefix-preserving ESAE] *Let $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a length- and prefix-preserving symmetric encryption scheme whose message space $\mathsf{MsgSp}_{\mathcal{SE}}$ contains messages of multiple of block-length n and let d be the maximum possible number of blocks (hereafter we denote the set of such strings by $D_{d,n}$). Let $\mathsf{OPerm}_{d,n}$ denote the family of all length- and prefix-preserving permutations on $D_{d,n}$. Let $\perp(\cdot)$ denote the oracle that always returns $\perp$ and r denote a random n-bit block (picked fresh each time it is encountered). For an adversary A with access to two oracles define the experiments:*

Experiment $\mathbf{Exp}^{\text{pp-0}}_{\mathcal{SE},A}$	Experiment $\mathbf{Exp}^{\text{pp-1}}_{\mathcal{SE},A}$
$sk \xleftarrow{\$} \mathcal{K}\ ;\ d \xleftarrow{\$} A^{\mathcal{E}(sk,\cdot),\mathcal{D}(sk,\cdot)}$	$g \xleftarrow{\$} \mathsf{OPerm}_{d+1,n}\ ;\ d \xleftarrow{\$} A^{g(\cdot\|\|r),\perp(\cdot)}$
Return d	Return d

We call A a pp-adversary *if it never repeats queries, never queries a response from its first oracle to its second, and all queries to its first oracle belong to $D_{d,n}$ and queries to its second belong to $D_{d+1,n}$. The pp-advantage of a A is defined as follows:*

$$\mathbf{Adv}^{\text{pp}}_{\mathcal{SE},A} \quad = \quad \Pr[\,\mathbf{Exp}^{\text{pp-0}}_{\mathcal{SE},A} = 0\,] - \Pr[\,\mathbf{Exp}^{\text{pp-1}}_{\mathcal{SE},A} = 0\,].$$

The scheme $\mathcal{SE}$ is said to be pp-secure *if for every efficient pp-adversary A the probability $\mathbf{Adv}^{\text{pp}}_{\mathcal{SE},B}$ is sufficiently small.* ▮

[4]Of course, one can always achieve authenticity using a MAC on top of the encryption scheme, but the point is that this would be excessive in some applications.

DISCUSSION. Analogous to the case of exact-match queries (see Section 4, discussion), our security definition here ensures that the server cannot learn anything about the data except which attribute values share a same prefix, which is obviously unavoidable in this context, where the granularity of such prefix-correlation is given by the length of the block cipher used in our construction below (and on the other hand it is bit-wise for the less-efficient, no-authenticity scheme of [43]). Here one has to be wary of frequency-based (in terms how many *distinct* plaintexts with a given prefix occur in the database) deduction of some prefixes when using text data, which may require adding bogus data to balance these frequencies. We stress that this analysis holds *only* in the presence of prefix-matching (or exact-match) queries. In a generalization and refinement of the approach of [32] that we present in Appendix 7, we show that our scheme *can* be used to efficiently support range-query processing as well but our security analysis is much more delicate.

OUR CONSTRUCTION AND ANALYSIS. As in [7], appealing constructions such as the authenticated encryption scheme OCB [39] with fixed IV can be shown insecure under Definition 6.1. We design a prefix-preserving ESAE scheme based on an interesting modification of the HPCBC cipher [7, Construction 8.1] that appends an all-zero block to a message to encrypt and uses a different block cipher on this last block to also achieve ciphertext-integrity, which may also be of independent interest.[5] It is efficient and uses one block cipher and one hash function operation per one block of input.

Definition 6.2 *Let $E\colon \{0,1\}^{ek} \times \{0,1\}^n \to \{0,1\}^n$ be a block cipher. Let $H\colon \{0,1\}^{hk} \times \{0,1\}^{2n} \to \{0,1\}^n$ be a family of functions. We associate to them a prefix-preserving ESAE scheme $\mathsf{HPCBC}^+ = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ defined as follows. The key generation algorithm chooses randomly a key $eK\|eK'\|hK$ where eK, eK' are (independent) keys for E and hK is a key for H. The encryption and decryption algorithms are defined as follows:*

```
Algorithm $\mathcal{E}(eK\|eK'\|hK, m)$
{ Parse $m$ as $m[1]\ldots m[l]$
$C[0] \leftarrow 0^n$ ; $m[0] \leftarrow 0^n$
For $i = 1, \ldots, l$ do
    $R \leftarrow m[i-1]||C[i-1]$
    $P[i] \leftarrow H(hK, R) \oplus m[i]$
    $C[i] \leftarrow E(eK, P[i]) \oplus H(hK, R)\}$
$R \leftarrow m[l]\|C[l]$
$P[l+1] \leftarrow H(hK, R) \oplus 0^n$
$C[l+1] \leftarrow E(eK', P[l+1]) \oplus H(hK,R)$
Return $C[1]\ldots C[l+1]$
```

```
Algorithm $\mathcal{D}(eK\|hK, C)$
{Parse $C$ as $C[1]\ldots C[l+1]$ with $l \geq 1$
$C[0] \leftarrow 0^n$ ; $m[0] \leftarrow 0^n$
For $i = 1, \ldots, l$ do
    $R \leftarrow m[i-1]||C[i-1]$
    $P[i] \leftarrow E^{-1}(eK, C[i] \oplus H(hK, R))$
    $m[i] \leftarrow H(hK, R) \oplus P[i]\}$
$R \leftarrow m[l]\|C[l]$
$P[l+1] \leftarrow E^{-1}(eK', C[l+1] \oplus H(hK, R))$
$m[l+1] \leftarrow H(hK, R) \oplus P[l+1]$
If $m[l+1] = 0^n$ then return $m[1]\ldots m[l+1]$
Else return $\perp$
```

We note that the 6 first lines of the algorithms (i.e. the part between braces) could be expressed more compactly as $C[1]\ldots C[l] \leftarrow \mathsf{HPCBC}(eK\|hK, m)$ and $m[1]\ldots m[l] \leftarrow \mathsf{HPCBC}^{-1}(eK\|hK, C)$. This explicit description of HPCBC is given here for completeness. To see the benefit of using our construction over plain HPCBC note that encryption along with a separate MAC (e.g. CMAC) to additionally achieve integrity would roughly double the computation time as compared to our construction.

Security of the scheme is based on the security of the underlying block cipher and the hash function. The corresponding definitions of PRP-CCA security of a block cipher and of almost-xor-universal hash functions is recalled in [7]. AES is believed to be PRP-CCA, and [7] provide references for secure hash function constructions. The proof (that also contains concrete security results) of the following theorem is in Appendix A.

Theorem 6.3 *Let $E\colon \{0,1\}^{ek} \times \{0,1\}^n \to \{0,1\}^n$ be a block cipher that is a PRP-CCA. and let $H\colon \{0,1\}^{hk} \times \{0,1\}^{2n} \to \{0,1\}^n$ be an almost-xor-universal family of hash functions. Then HPCBC^+ defined via Definition 6.2 is a pp-secure prefix-preserving ESAE scheme.*

[5]In fact our construction treats HPCPC as a black-box so any on-line cipher that is OPRP-CCA (see [7] for the definition) can be used, but we suggest HPCBC for concreteness.

7 On Efficient Range-Query Processing from Prefix-Preserving Schemes

In [32] it is shown that encrypting data via a bit-wise prefix-preserving scheme allows efficient (as opposed to scanning the whole database) range queries over the data by specifying the possible prefixes for a desired range. Introducing our prefix-preserving ESAE as well provides a generalized approach, where the block size is not just one bit but a variable parameter. It was shown in [32] that certain attacks are possible if their scheme is used for range queries. We leave it to the future work to generalize such attacks and discuss what is the best level of security prefix-preserving schemes can provide.

8 Conclusions and Open Problems

We have presented new security models and the first practical, provably-secure constructions to support several basic query types one would like for outsourced databases. While as noted in Section 4 our schemes naturally support outsourcing counting operations over the data, it would be desirable to also outsource other types of aggregate queries, e.g. sum and average, as well. The latter is challenging in the symmetric setting due to the lack of homomorphic encryption schemes. An approach explored in [35] is to instead include some other "aggregate" information with the ciphertexts but it remains to develop a proper understanding of security for such schemes.

References

[1] The final HIPAA security rule. *Federal Register. Available at* `http://www.hipaadvisory.com/regs/finalsecurity/index.htm`, 2003.

[2] G. Aggarwal, M. Bawa, P. Ganesan, H. Garcia-Molina, K. Kenthapadi, R. Motwani, U. Srivastava, D. Thomas, and Y. Xu. Two can keep a secret: A distributed architecture for secure database services. In *CIDR*, 2005.

[3] R. Agrawal, J. Kiernan, R. Srikant, and Y. Xu. Order preserving encryption for numeric data. In *SIGMOD*, 2004.

[4] J.-H. An, Y. Dodis, and T. Rabin. On the security of joint signature and encryption. In *EUROCRYPT*, 2002.

[5] M. Bellare. Practice-oriented provable-security. In *Information Security Workshop, ISW*, 1997.

[6] M. Bellare. New proofs for NMAC and HMAC: Security without collision-resistance. CRYPTO, 2006.

[7] M. Bellare, A. Boldyreva, L. R. Knudsen, and C. Namprempre. Online ciphers and the Hash-CBC construction. In *CRYPTO*, 2001.

[8] M. Bellare, A. Boldyreva, and A. O'Neill. Efficiently-searchable and deterministic asymmetric encryption. Cryptology ePrint Archive, Report 2006/186, 2006. `http://eprint.iacr.org/2006/186/`.

[9] M. Bellare, R. Canetti, and H. Krawczyk. Keying hash functions for message authentication. In *CRYPTO*, 1996.

[10] M. Bellare, A. Desai, E. Jokipii, and P. Rogaway. A concrete security treatment of symmetric encryption. In *FOCS*, 1997.

[11] M. Bellare, T. Kohno, and C. Namprempre. Authenticated encryption in SSH: provably fixing the SSH binary packet protocol. In *CCS*. ACM Press, 2002.

[12] M. Bellare and C. Namprempre. Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. In *ASIACRYPT 2000*, 2000.

[13] M. Bellare and P. Rogaway. The game-playing technique and its application to triple encryption. In *EUROCRYPT*, 2006.

[14] J. Black and P. Rogaway. CBC MACs for arbitrary-length messages: The three-key constructions. In *CRYPTO*, 2000.

[15] D. Boneh, G. Di Crescenzo, R. Ostrovsky, and G. Persiano. Public key encryption with keyword search. In *EUROCRYPT*, 2004.

[16] R. Canetti, H. Krawczyk, and J. Nielsen. Relaxing chosen-ciphertext security. In *CRYPTO*, 2003.

[17] A. Ceselli, E. Damiani, S. De Capitani di Vimercati, S. Jajodia, S. Paraboschi, and P. Samarati. Modeling and assessing inference exposure in encrypted databases. *ACM Trans. Inf. Syst. Secur.*, 8(1):119–152, 2005.

[18] Y.-C. Chang and M. Mitzenmacher. Privacy preserving keyword searches on remote encrypted data. In *ACNS*, 2005.

[19] S. Contini and Y.-L. Yin. Forgery and partial key-recovery attacks on HMAC and NMAC using hash collisions. Cryptology ePrint Archive, Report 2006/319, 2006. `http://eprint.iacr.org/`.

[20] R. Curtmola, J. Garay, S. Kamara, and R. Ostrovsky. Searchable symmetric encryption: Improved definitions and efficient constructions. Cryptology ePrint Archive, Report 2006/210, 2006.

[21] E. Damiani, S. De Capitani di Vimercati, S. Paraboschi, and P. Samarati. Computing range queries on obfuscated data. In *Information Processing and Management of Uncertainty in Knowledge-Based Systems*, 2004.

[22] E. Damiani, S. De Capitani Vimercati, S. Jajodia, S. Paraboschi, and P. Samarati. Balancing confidentiality and efficiency in untrusted relational DBMSs. In *CCS*, 2003.

[23] E.-J. Goh. Secure indexes. Cryptology ePrint Archive, Report 2003/216, 2003. `http://eprint.iacr.org/2003/216/`.

[24] S. Goldwasser and S. Micali. Probabilistic encryption. In *Journal of Computer and Systems Sciencies*, volume 28, pages 270–299, 1984.

[25] P. Golle, J. Staddon, and B. Waters. Secure conjunctive keyword search over encrypted data. In *Applied Cryptography and Network Security Conference*.

[26] H. Hacigümüs, B. Iyer, C. Li, and S. Mehrotra. Executing SQL over encrypted data in the database-service-provider model. In *SIGMOD*, 2002.

[27] H. Hacigümüs, B. R. Iyer, and S. Mehrotra. Efficient execution of aggregation queries over encrypted relational databases. In *DASFAA*, pages 125–136, 2004.

[28] B. Hore, S. Mehrotra, and G. Tsudik. A privacy-preserving index for range queries. In *VLDB*, pages 720–731, 2004.

[29] B. R. Iyer, S. Mehrotra, E. Mykletun, G. Tsudik, and Y. Wu. A framework for efficient storage security in RDBMS. In *EDBT*, pages 147–164, 2004.

[30] M. Kantracioglu and C. Clifton. Security issues in querying encrypted data. In *DBSec*, 2005.

[31] F. Li, M. Hadjieleftheriou, G. Kollios, and L. Reyzin. Dynamic authenticated index structures for outsourced databases. In *SIGMOD*. ACM Press, 2006.

[32] J. Li and E. Omiecinski. Efficiency and security trade-off in supporting range queries on encrypted databases. In *DBSec*, pages 69–83, 2005.

[33] E. Mykletun, M. Narasimha, and G. Tsudik. Authentication and integrity in outsourced databases. In *NDSS*, 2004.

[34] E. Mykletun and G. Tsudik. Incorporating a secure coprocessor in the database-as-a-service model. In *International Workshop on Innovative Architecture for Future Generation High Performance Processors and Systems*, 2005.

[35] E. Mykletun and G. Tsudik. Aggregation queries in the database-as-a-service model. In *DBSEC*, 2006.

[36] M. Narasimha and G. Tsudik. DSAC: integrity for outsourced databases with signature aggregation and chaining. In *CIKM*, pages 235–236, 2005.

[37] M. Narasimha and G. Tsudik. Authentication of outsourced databases using signature aggregation and chaining. In *DASFAA*, pages 420–436, 2006.

[38] G. Özsoyoglu, D. A. Singer, and S. S. Chung. Anti-tamper databases: Querying encrypted databases. In *DBSec*, pages 133–146, 2003.

[39] P. Rogaway, M. Bellare, J. Black, and T. Krovetz. OCB: a block-cipher mode of operation for efficient authenticated encryption. In *ACM CCS '01*, 2001.

[40] P. Rogaway and T. Shrimpton. A provable-security treatment of the key-wrap problem. In *EUROCRYPT*, 2006.

[41] Arsenal Digital Solutions. Top 10 reasons to outsource remote data protection. *`http://www.arsenaldigital.com/services/remote_data_protection.htm`*.

[42] D. X. Song, D. Wagner, and A. Perrig. Practical techniques for searches on encrypted data. In *IEEE Symposium on Security and Privacy*, pages 44–55, 2000.

[43] J. Xu, J. Fan, M. H. Ammar, and S. B. Moon. Prefix-preserving IP address anonymization: Measurement-based security evaluation and a new cryptography-based scheme. In *ICNP*, 2002.

A Proofs.

A.1 Proof of Theorem 5.2.

Here are the adversaries for the proof.

```
**Adversary** $B_1^{\mathcal{M}(sk,\mathcal{LR}(\cdot,\cdot,b))}$
$sk_M \xleftarrow{\$} \mathcal{K}_M$ ; $sk_E \xleftarrow{\$} \mathcal{K}_E$ ; $b' \xleftarrow{\$} \{0,1\}$
Run $A$ twice (independently),
replying to its oracle queries as follows:
    On lr-encryption query $(m_0, m_1)$:
First time:
    $\sigma \xleftarrow{\$} \mathcal{M}(sk_M, m_0)$ ; $C \leftarrow \mathcal{E}(sk_E, m_{b'})$
    Return $\sigma \| C$
Second time:
    $\sigma \xleftarrow{\$} \mathcal{M}(sk, \mathcal{LR}(m_0, m_1, b))$ ; $C \leftarrow \mathcal{E}(sk_E, m_{b'})$
    Return $\sigma \| C$
    On decryption query $Y$:
Return $\perp$
Let $d_0, d_2$ be the respective outputs of $A$
If $d_2 = d_0$ then return 0 else return 1
```

```
Adversary $B_2^{\mathcal{M}(sk,\cdot),\mathcal{V}(sk,\cdot,\cdot)}$
$sk_E \xleftarrow{\$} \mathcal{K}_E$ ; $b \xleftarrow{\$} \{0,1\}$ ; $ctr \leftarrow 0$ ; $n \xleftarrow{\$} \{1, \ldots q_d\}$
Run $A$, replying to its oracle queries as follows:
    On lr-encryption query $(m_0, m_1)$:
    $\sigma \xleftarrow{\$} \mathcal{M}(sk, m_b)$ ; $C \leftarrow \mathcal{E}(sk_E, m_b)$ Return $\sigma \| C$
    On decryption query $Y$:
    Parse $Y$ as $\sigma \| C$ ; $ctr \leftarrow ctr + 1$
    If $ctr = n$ then abort and return $(\mathcal{D}(sk_E, C), \sigma)$
    Else return $\perp$
```

```
Adversary $B_3^{\mathcal{E}(sk,\mathcal{LR}(\cdot,\cdot,b))}$
$sk \xleftarrow{\$} \mathcal{K}_M$
Run $A$, replying to its oracle queries as follows:
    On lr-encryption query $(m_0, m_1)$:
    $C \xleftarrow{\$} \mathcal{E}(sk, \mathcal{LR}(m_0, m_1, b))$ ; $\sigma \leftarrow \mathcal{M}(sk_M, m_0)$ Return $\sigma \| C$
    On decryption query $Y$: Return $\perp$
Let $d$ be the output of $A$, return $d$
```

We claim they satisfy the given relation. To establish the claim, we use the code-based game-playing proof technique in the style of [13]. Following [8], let us first recall some fundamentals of this technique.

A game consists of an Initialize procedure, procedures that respond to adversary oracle queries, and a Finalize procedure. Below we present a total of four games. Games $G_1 - G_3$ have the same Initialize procedure, this being the one shown with the boxed statement included, then this procedure for game G_4 has this boxed statement removed. The procedure to respond to lr-encryption queries is the same for all games, which is shown just below the former. The procedure to respond to decryption queries includes the boxed statement in the given code for game G_1, then omits it for the remaining games. The finalize procedure includes the boxed statement in the top right code for games G_1, G_2, then drops it for games G_3, G_4.

```
procedure Initialize        Games $G_1 - G_3/G_4$
$sk_M \xleftarrow{\$} \mathcal{K}_M$ ; $sk_E \xleftarrow{\$} \mathcal{K}_E$
$b, b' \xleftarrow{\$} \{0,1\}$ If $b \neq b'$ then $\boxed{b' \leftarrow b}$

On lr-encryption query $(m_0, m_1)$:        All Games
$\sigma \leftarrow \mathcal{M}(sk_M, m_{b'})$ ; $C \xleftarrow{\$} \mathcal{E}(sk_E, m_b)$
Return $\sigma \| C$
```

```
On decryption query $Y$:        Game $G_1/G_2 - G_4$
Parse $Y$ as $\sigma \| C$ ; $m \leftarrow \mathcal{D}(sk_E, C)$
If $\mathcal{M}(sk_M, m) = \sigma$ then
    bad $\leftarrow$ true ; $\boxed{\text{Return } m}$ else return $\perp$

procedure Finalize($d$):        Game $G_1, G_2/G_3, G_4$
If $m \neq \perp$ was decrypted $\boxed{d \leftarrow b}$
Return $d$
```

We will be executing A with each of these games. The execution of A with G_i is determined as follows. First, the Initialize procedure executes. Variables set in this procedure are global to the rest of the code. Now the adversary A executes, its lr-encryption and decryption oracle queries being answered by the procedures for this purpose associated to G_i. The output d of A becomes the input to the Finalize procedure of G_i. The output of the game is whatever is returned by the Finalize procedure. We let "$G_i^A \Rightarrow b$" denote the event that the output of Game G_i, when executed with A, is the bit b chosen at random in the Initialize procedure.

Equation (1) follows from the following sequence of inequalities, which it remains to justify:

$$\begin{aligned}
\frac{1}{2}+\frac{1}{2}\mathbf{Adv}^{\text{ind-esae}}_{\mathcal{SE}^*,A} &= \Pr[\,G_1^A \Rightarrow b\,] \le \Pr[\,G_2^A \Rightarrow b\,] + \Pr[G_1^A \text{ sets } \mathsf{bad}\,] && (2)\\
&\le \Pr[\,G_2^A \Rightarrow b\,] + q_d\mathbf{Adv}^{\text{uf-cma}}_{\mathcal{MAC},B_2} \le \Pr[\,G_3^A \Rightarrow b\,] + q_d\mathbf{Adv}^{\text{uf-cma}}_{\mathcal{MAC},B_2} && (3)\\
&\le \Pr[\,G_4^A \Rightarrow b\,] + q_d\mathbf{Adv}^{\text{uf-cma}}_{\mathcal{MAC},B_2} + \mathbf{Adv}^{\text{pp-mac}}_{\mathcal{MAC},B_1} && (4)\\
&\le \frac{1}{2}+\frac{1}{2}\mathbf{Adv}^{\text{ind-cpa}}_{\mathcal{SE},B_3} + q_d\mathbf{Adv}^{\text{uf-cma}}_{\mathcal{MAC},B_2} + \mathbf{Adv}^{\text{pp-mac}}_{\mathcal{MAC},B_1} && (5)
\end{aligned}$$

The advantage $\mathbf{Adv}^{\text{ind-esae}}_{\mathcal{MAC},B}$, defined as the difference in the probabilities that two experiments return 1, is, as usual, also equal to $2p-1$ where p is the probability that the adversary correctly guesses the challenge bit b in a game where we pick b at random and run the adversary with the first experiment if $b=1$ and the second if $b=0$. Game G_1 is exactly this game written in a convenient way. We have justified Equation (2).

Equation (2) follows from the Fundamental Lemma in [13], which says that difference in the probabilities that two identical-until-bad games return 1 is bounded by the probability that bad is set. Consider the uf-cma adversary B_2 presented in above. As in the proof of the Fundamental Lemma in [13], we can consider a common finite set of coins associated with the executions of B_2 and A with G_1. Suppose that A is executed with G_1 with a particular sequence of coins Coins from this set. If bad is set, it means that A has produced a mac for m as defined in the procedure to respond to decryption queries, just as it would have in the uf-cma experiment (recall that we disallow queries with the same tag, and that the MAC is deterministic, meaning there is one tag per message). Thus when A is executed with Coins as a subroutine of B_2, if n has the correct ordinal number for this query then B_2 outputs a valid forgery for $\mathcal{MAC}$. Equation (3) then follows from taking probabilities over the random choice of Coins as well as the (independent) coins used in the overlying experiments.

Now notice that when G_2 is executed, $\perp$ can never be returned in the procedure to respond to decryption oracle queries. Thus we can drop the boxed code there without affecting the distribution of its output, resulting the equivalent game we call G_3. This justifies Equation (3).

Next we are interested in $\Pr[G_3^A \Rightarrow b] - \Pr[G_4^A \Rightarrow b]$. As before, let us compare the execution of A with G_3 to the pp-mac adversary B_1 presented above. Let "$G_4^{A,i,j} \Rightarrow b$" denote the event that A outputs $b \in \{0,1\}$ when run with G_4 hardwired with $b=i$ and $b'=j$ and $b=i$ denote the event that game G_4 sets the bit b to i during its execution with A. Then we have:

$$\begin{aligned}
\Pr[G_3^A \Rightarrow b] - \Pr[G_4^A \Rightarrow b] &\le \Pr[b=0 \,\wedge\, b'=1](\Pr[G_4^{A,0,0} \Rightarrow 0] - \Pr[G_4^{A,0,1} \Rightarrow 0] + \Pr[G_4^{A,0,0} \Rightarrow 1]\\
&- \Pr[G_4^{A,0,1} \Rightarrow 1]) + \Pr[b=1 \,\wedge\, b'=0]([G_4^{A,1,1} \Rightarrow 0] - \Pr[G_4^{A,1,0} \Rightarrow 0]\\
&+ \Pr[G_4^{A,1,1} \Rightarrow 1] - \Pr[G_4^{A,1,0} \Rightarrow 1]).
\end{aligned}$$

The above is a standard conditioning argument, expanding $\Pr[G_4^A \Rightarrow b] - \Pr[G_3^A \Rightarrow b]$ accordingly. Similarly, comparing the code of game G_4 and adversary B_1 and working from the opposite direction, we can get:

$$\begin{aligned}
\mathbf{Adv}^{\text{pp-mac}}_{\mathcal{MAC},B_1} &\ge \Pr[b=0](\Pr[G_4^{A,0,0} \Rightarrow 0]^2 + \Pr[G_4^{A,0,0} \Rightarrow 1]^2) + \Pr[b=1](\Pr[G_4^{A,1,0} \Rightarrow 0]^2 + \Pr[G_4^{A,1,0} \Rightarrow 1]^2)\\
&- \Pr[b=0](\Pr[G_4^{A,0,0} \Rightarrow 0]\Pr[G_4^{A,0,1} \Rightarrow 0] + \Pr[G_4^{A,0,0} \Rightarrow 1]\Pr[G_4^{A,0,1} \Rightarrow 1])\\
&- \Pr[b=1](\Pr[G_4^{A,1,0} \Rightarrow 0]\Pr[G_4^{A,1,1} \Rightarrow 0] + \Pr[G_4^{A,1,0} \Rightarrow 1]\Pr[G_4^{A,1,1} \Rightarrow 1])\\
&\ge \Pr[b=0]\frac{1}{2}(\Pr[G_4^{A,0,0} \Rightarrow 0] - \Pr[G_4^{A,0,1} \Rightarrow 0]) + \Pr[b=0]\frac{1}{2}(\Pr[G_4^{A,0,0} \Rightarrow 1] - \Pr[G_4^{A,0,1} \Rightarrow 1])\\
&+ \Pr[b=1]\frac{1}{2}(\Pr[G_4^{A,1,0} \Rightarrow 0] - \Pr[G_4^{A,1,1} \Rightarrow 0]) + \Pr[b=1]\frac{1}{2}(\Pr[G_4^{A,1,0} \Rightarrow 1] - \Pr[G_4^{A,1,1} \Rightarrow 1]),
\end{aligned}$$

where the last inequality follows from the fact that if, e.g., the events $G_4^{A,0,0} \Rightarrow 0, G_4^{A,0,0} \Rightarrow 1$ do not take the uniform distribution, then $\Pr[G_4^{A,0,0} \Rightarrow 0](\Pr[G_4^{A,0,0} \Rightarrow 0] - \Pr[G_4^{A,0,1} \Rightarrow 0]) + \Pr[G_4^{A,0,0} \Rightarrow 1](\Pr[G_4^{A,0,0} \Rightarrow 1] - \Pr[G_4^{A,0,1} \Rightarrow 1])$ can only go up. Using independence of setting b, b' from the other events in question, we deduce that $\Pr[G_3^A \Rightarrow b] - \Pr[G_4^A \Rightarrow b] \leq \mathbf{Adv}^{\text{pp-mac}}_{\mathcal{MAC},B_1}$ as desired to justify Equation (4).

Finally, Equation (5) then follows easily from comparing the code of G_4 to that of the ind-cpa adversary B_3. ▮

A.2 Proof of Theorem 5.5.

Let A be an esae adversary against $\mathcal{SE}$ making at most q_e encryption queries and q_d decryption queries. Then we claim there exists an ind-cpa adversary B against $\mathcal{SE}$ such that:

$$\mathbf{Adv}^{\text{ind-esae}}_{\mathcal{SE}^*,A} \quad \leq \quad \mathbf{Adv}^{\text{ind-cpa}}_{\mathcal{SE},B} + 2\mathbf{Adv}^{\text{prf}}_{\mathcal{MAC}} + \frac{2q_d}{mc_{\mathcal{SE}}},$$

which implies the theorem. As usual in reduction-based security proofs, since we assume that $\mathcal{MAC}$ is a PRF, we can first substitute $\mathcal{MAC}$ in the construction with a truly random function for the analysis. Hence the analysis in the proof of Theorem 4.3 in [8] shows an esae adversary against $\mathcal{SE}^*$ making at most q_d decryption queries has at most a $q_d/mc_{\mathcal{SE}^*}$ chance of querying a new valid ciphertext to its decryption oracle, and, as an esae adversary is not allowed to repeat components of its encryption queries, the following ind-cpa adversary B against $\mathcal{SE}$ otherwise perfectly simulates for any esae adversary A against $\mathcal{SE}^*$:

```
Adversary $B^{\mathcal{E}(sk,\mathcal{LR}(\cdot,\cdot,b))}$
    Run $A$, answering its oracle queries as follows:
    On lr-encryption query $(m_0, m_1)$ return $\mathcal{E}^*(sk, \mathcal{LR}(m_0, m_1, b))$
    On decryption query $C$: Return $\perp$
    Until $A$ halts with output $d$
    Return $d$
```

We use here that $\mathcal{MAC}$ is a truly random function since $\mathcal{E}$ uses truly random coins to encrypt, whereas $\mathcal{E}^*$ (which A is expecting) uses the mac of the message as the coins. By a game-playing argument (see Appendix **??**), we can conclude the claim. ▮

A.3 Proof of Theorem 6.3.

Let B be an odae-adversary against HPCBC^+ making at most q_e encryption queries and q_d decryption queries of at most d blocks of length n each. We prove that:

$$\begin{aligned}\mathbf{Adv}^{\text{pp}}_{\mathsf{HPCBC}^+,A} \quad &\leq \quad \mathbf{Adv}^{\text{prp-cca}}_{E} + \mathbf{Adv}^{\text{oprp-cpa}}_{\mathsf{HPCBC}} \\ &+ \quad \left(\frac{1}{1 - \frac{q_e(q_e-1)}{2^n}} + \frac{1}{1 - \frac{q_d(q_d-1)}{2^n}} \right) \mathbf{Adv}^{\text{axu}}_{H} + \frac{q_e(q_e-1) + q_d}{2^{n-1}},\end{aligned} \tag{6}$$

which implies the theorem.

The proof uses ideas from the proof of Theorem 8.3 in [7]. For $1 \leq j, j' \leq q$, we say that query a query $m_{j'}$ is l-*trivial* (the negation being l-nontrivial) if there exists some query m_j with $j < j'$ such that $nl \leq |\mathsf{LCP}_n(m_j, m_{j'})|$.

For the proof, we employ the code-based game-playing proof technique of [13]. Consider the following games associated with the execution of B shown in Figure 1.

First observe that $\mathbf{Adv}^{\text{pp}}_{\mathsf{HPCBC}^+,B} = \Pr[\, G_1^A \Rightarrow 0\,] - \Pr[\, G_6^A \Rightarrow 0\,]$, by noting that the difference between game G_6 here and the "ideal" experiment in Definition 6.1 is that for latter, what we call block $C[l+1]$ in G_6 actually takes value $g(C[1]\| \ldots \|C[l]\|r)$, but since r is picked anew at random there and A never gets any information about it, by Proposition 3.5 in [7] $m[l+1]$ has the same distribution in either case. Now a standard substitution

argument gives us
$\Pr[\,G_1^A \Rightarrow 0\,] \leq \Pr[\,G_2^A \Rightarrow 0\,] + \mathbf{Adv}_E^{\mathrm{prp\text{-}cca}} + \mathbf{Adv}_{\mathsf{HPCBC}}^{\mathrm{oprp\text{-}cpa}}$ and by the Fundamental Lemma of Game Playing we also have that $\Pr[\,G_2^A \Rightarrow 0\,] \leq \Pr[\,G_4^A \Rightarrow 0\,] + \Pr[\,G_2^A \text{ sets } \mathsf{bad}_0\,] + \Pr[\,G_3^A \text{ sets } \mathsf{bad}_1\,]$. On the other hand, the lemma also yields
$\Pr[\,G_6^A \Rightarrow 0\,] \geq \Pr[\,G_5^A \Rightarrow 0\,] - \Pr[\,G_5^A \text{ sets } \mathsf{bad}\,]$, and by way of Claim 8.6 in [7] we find that $\Pr[\,G_4^A \Rightarrow 0\,] - \Pr[\,G_5^A \Rightarrow 0\,] \leq 0$. Putting this all together, we deduce:

$$\begin{aligned}
\mathbf{Adv}_{\mathsf{HPCBC}^+,B}^{\mathrm{pp}} &= \Pr[\,G_1^A \Rightarrow 0\,] - \Pr[\,G_6^A \Rightarrow 0\,] \\
&\leq \Pr[\,G_4^A \Rightarrow 0\,] - \Pr[\,G_5^A \Rightarrow 0\,] + \mathbf{Adv}_E^{\mathrm{prp\text{-}cca}} \\
&+ \mathbf{Adv}_{\mathsf{HPCBC}}^{\mathrm{oprp\text{-}cpa}} + \Pr[\,G_2^A \text{ sets } \mathsf{bad}_0\,] + \Pr[\,G_3^A \text{ sets } \mathsf{bad}_1\,] \\
&\leq \Pr[\,G_4^A \Rightarrow 0\,] - \Pr[\,G_5^A \Rightarrow 0\,] + \Pr[\,G_5^A \text{ sets } \mathsf{bad}\,] \\
&+ \mathbf{Adv}_E^{\mathrm{prp\text{-}cca}} + \mathbf{Adv}_{\mathsf{HPCBC}}^{\mathrm{oprp\text{-}cpa}} + \Pr[\,G_2^A \text{ sets } \mathsf{bad}_0\,] \\
&+ \Pr[\,G_3^A \text{ sets } \mathsf{bad}_1\,] \\
&\leq \mathbf{Adv}_E^{\mathrm{prp\text{-}cca}} + \mathbf{Adv}_{\mathsf{HPCBC}}^{\mathrm{oprp\text{-}cpa}} \\
&+ \Pr[\,G_2^A \text{ sets } \mathsf{bad}_0\,] + \Pr[\,G_3^A \text{ sets } \mathsf{bad}_1\,] + \Pr[\,G_5^A \text{ sets } \mathsf{bad}\,]
\end{aligned}$$

It remains to bound the probability that bad is set for the various terms above. First we look at game G_5 and ask the probability that bad is set there when executed with A. For the top case (meaning the first point in the code at which bad can be set) observe that if the decryption query C is l-nontrivial then $C[l+1]$ is random and independent of everything else, thus by the birthday bound the total probability that bad is set in this top case is at most $q_e(q_e-1)/2^n$ (recall the blocks are of size n). By analogous reasoning, in the bottom case the bound is $q_d/2^n$, thus taking a union bound of both we have $\Pr[\,G_4^A \text{ sets } \mathsf{bad}\,] \leq (q_e(q_e-1)+q_d)/2^n$.

Next we consider game G_2 and bound the probability bad_0 is set when executed with A in the top case here as follows. Prop. 3.5 of [7] and the the birthday bound tell us the probability that for all l-nontrivial queries m that A makes to its encryption oracle the corresponding blocks $C[l]$ are distinct is at least $1-(q_e(q_e-1)/2^n)$. Now assuming they are all distinct, if the blocks $P[l+1]$ agree across any two such queries this violates the almost-xor-universality of H since the corresponding values of R in its input must differ. In other words, $\Pr[\,G_2^A \text{ sets } \mathsf{bad}_0\,] \leq (1/(1-(q_e(q_e-1)/2^n)) + 1/(1-(q_d(q_d-1)/2^n)))\mathbf{Adv}_H^{\mathrm{axu}}$, where we use symmetry of the construction and a union bound; the adversary works analogously to that for Claim 8.9 in [7].

Finally, using the same justification as above (for when G_5 sets bad), we have that $\Pr[\,G_3^A \text{ sets } \mathsf{bad}_1\,] \leq (q_e(q_e-1)+q_d)/2^n$, which fills in the remaining term as desired; combining like-terms and simplifying gives Equation (6). ▌

```
**Game** $G_1$
**procedure Initialize**:
$eK, eK' \xleftarrow{\$} \{0,1\}^{ek}$ ; $hK \xleftarrow{\$} \{0,1\}^{hk}$
$g \xleftarrow{\$} \mathsf{OPerm}_{d+1,n}$
**On encryption query** $m$:
$C[1] \dots C[l] \leftarrow \mathsf{HPCBC}(eK \| hK, m)$
$R \leftarrow m[l] \| C[l]$
$P[l+1] \leftarrow H(hK, R) \oplus 0^n$
$C[l+1] \leftarrow E(eK', P[l]) \oplus H(hK,R)$
Return $C[1] \dots C[l+1]$
**On decryption query** $C$:
Parse $C$ as $C[1] \dots C[l+1]$ with $l \geq 1$
$m[1] \dots m[l] \leftarrow \mathsf{HPCBC}^{-1}(eK \| hK, C[1] \dots C[l])$
$R \leftarrow m[l] \| C[l]$
$P[l+1] \leftarrow E^{-1}(eK', C[l+1] \oplus H(hK, R))$
$m[l+1] \leftarrow H(hK, R) \oplus P[l+1]$
If $m[l+1] = 0^n$ then return $m[1] \dots m[l+1]$
Else return $\perp$
**procedure Finalize**(d) :
If $m \neq \perp$ was decrypted then
    $d \leftarrow 0$
Return $d$
```

```
**Game** $G_5/G_6$
**procedure Initialize**:
$L \leftarrow \emptyset$
$g' \xleftarrow{\$} \mathsf{OPerm}_{d+1,n}$ ; $g \xleftarrow{\$} \mathsf{OPerm}_{d,n}$
**On encryption query** $m$:
$C[1] \dots C[l] \leftarrow g(m)$
$C'[1] \| \dots \| C'[l] \| C[l+1] \leftarrow g'(C[1] \| \dots \| C[l] \| 0^n)$
If $C[l+1] \in L$ and $C$ is $l$-nontrivial then
bad $\leftarrow$ true ; $\boxed{\text{abort and return 1}}$
$L \leftarrow L \cup C[l+1]$
Return $C[1] \dots C[l+1]$
**On decryption query** $C$:
$m[1] \dots m[l+1] \leftarrow g'^{(-1)}(C)$
If $m[l+1] = 0^n$ then
    bad $\leftarrow$ true ; $\boxed{\text{abort and return 1}}$
Return $\perp$
**procedure Finalize**(d) :
Return $d$
```

```
**Game** $G_2/G_3/G_4$
**procedure Initialize**:
$L_P, L_C, L_Q \leftarrow \emptyset$
$hK \leftarrow \{0,1\}^{hk}$
$g \xleftarrow{\$} \mathsf{OPerm}_{d+1,n}$
$\pi \xleftarrow{\$} \mathsf{Perm}_n$ ; $b \xleftarrow{\$} \{0,1\}$
**On encryption query** $m$:
$C[1] \dots C[l] \leftarrow g(m)$
$R \leftarrow m[l] \| C[l]$
$P[l+1] \leftarrow H(hK, R) \oplus 0^n$
If $P[l+1] \in L_P$ and $m$ is $l$-nontrivial then
    bad$_0$ $\leftarrow$ true ; ADD: $\boxed{\text{abort and return 1}}$
$L_P \leftarrow L_P \cup P[l+1]$
$C[l+1] \leftarrow \pi(P[l]) \oplus H(hK,R)$
If $C[l+1] \in L_C$ and $m$ is $l$-nontrivial then
    bad$_1$ $\leftarrow$ true ; ADD: $\boxed{\boxed{\text{abort and return 1}}}$
$L_C \leftarrow L_C \cup C[l+1]$
Return $C[1] \dots C[l+1]$
**On decryption query** $C$:
Parse $C$ as $C[1] \dots C[l+1]$ with $l \geq 1$
$m[1] \dots m[l] \leftarrow g^{-1}(C[1] \dots C[l])$
$R \leftarrow m[l] \| C[l]$
$Q[l+1] \leftarrow C[l+1] \oplus H(hK, R)$
If $Q[l+1] \in L_Q$ and $C$ is $l$-nontrivial then
    bad$_0$ $\leftarrow$ true ; ADD: $\boxed{\text{abort and return 1}}$
$L_Q \leftarrow L_Q \cup Q[l+1]$
$P[l+1] \leftarrow \pi^{-1}(Q[l+1])$
$m[l+1] \leftarrow H(hK, R) \oplus P[l+1]$
If $m[l+1] = 0^n$ and $C$ is $l$-nontrivial then
    bad$_1$ $\leftarrow$ true ; ADD: $\boxed{\text{abort and return 1}}$
If $m[l+1] = 0^n$ then return $m[1] \dots m[l+1]$
Else return $\perp$
**procedure Finalize**(d) :
If $m \neq \perp$ was decrypted then
    $d \leftarrow 0$
Return $d$
```

Figure 1: Games for for the proof of Theorem 6.3. The games are determined as follows. The boxed statements are included in the games preceding a slash-mark, and removed for the games following it, unless the word “ADD:” appears before the box, in which case it is the opposite. Moreover, the number of boxes around a statement indicates the order in which these statements are added or removed. For example, in the Initialize procedure marked $G_2/G_3/G_4$, for game G_2 all boxed statements are absent, then for game G_3 the single-boxed statements only are added, then for game G_4 all boxed statements are present.