On Minimizing the Size of Encrypted Databases

Giovanni Di Crescenzo, David Shallcross

On Minimizing the Size of Encrypted Databases

Giovanni Di Crescenzo, David Shallcross

Applied Communication Sciences, Basking Ridge, NJ, USA.
E-mail: {gdicrescenzo,dshallcross}@appcomsci.com

Abstract. Motivated by applications to maintaining confidentiality and efficiency of encrypted data access in cloud computing, we uncovered an inherent confidentiality weakness in databases outsourced to cloud servers, even when encrypted. To address this weakness, we formulated a new privacy notion for outsourced databases and (variants of) a classical record length optimization problem, whose solutions achieve the new privacy notion. Our algorithmic investigation resulted in a number of exact and approximate algorithms, for arbitrary input distributions, and in the presence of record additions and deletions. Previous work only analyzed an unconstrained variant of our optimization problem for specific input distributions, with no attention to running time or database updates.

1 Introduction

As the cloud computing paradigm is entering many of today's distributed computing applications, the research community is investigating a host of associated problems in many areas, including privacy, security, and algorithmic efficiency. One central cloud computing capability consists of outsourcing data to servers in the cloud, in a way that delegates the management of the data to the cloud servers while allowing efficient data access from authorized clients. In a typical instantiation of this capability, the data owner publishes a searchable database in the cloud and clients make their ordinary database queries directly to the cloud server. Because of confidentiality requirements on the data, and of the often unknown location of cloud servers as well as unknown entities who closely manage them, the data owner typically chooses to publish an encrypted version of the database, which can be later queried using privacy-preserving database retrieval protocols, and subject to compliance to specific query-based access policies. Unfortunately, encryption alone does not suffice to protect data confidentiality in these contexts, as encryption is known to hide all partial information but the length of the encrypted data. Accordingly, a cloud server with side channel information about the length of individual database records can derive confidential information about the content of the encrypted database. (As an example from the finance industry, if stock fund prospectuses are longer than single stock prospectuses, a cloud server can detect the relative density of stock funds and/or single stocks in the database, even if encrypted).

Our contribution. Our approach to overcome these shortcomings consists of a suitable combination of padding short database records and splitting long database records so to normalize all records to have the same length, while still guaranteeing efficient access to them from clients. This approach calls for a new privacy model and a new optimization problem. In our privacy model, we require that the outsourced encrypted database

at most leaks a symmetric function of the original record lengths. Our optimization problem (a variant of a classical record length problem, studied in the statistical and computer memory management literature) is defined as follows: an encrypted database with a large number n of records of different sizes $s_1, \ldots, s_n \leq s_{max}$, needs to be normalized, via padding (e.g., adding a fixed string of a determined length) or splitting (e.g., dividing the record into 2 or more pieces and possibly padding again on the last piece) into a database with a potentially larger number n' of records, all having the same size σ. Padding obviously increases the database size, but so does splitting, as each record contains an a-bit searchable header, which has to be replicated on all pieces resulting from the record split. We want to find the (exactly or approximately) optimal σ so that the total size of the database (increased due to padding and splitting) is minimized under a constraint that bounds the increase in the maximum (or average) search time for a given record. For a cleaner problem formulation, applicable to any possible search strategy, we model this constraint by imposing an upper bound on the number of pieces derived from a split of any given record (or of all records, respectively).

Our exact algorithms perform $O(n \cdot s_{max})$ or $O(n + s_{max} \log s_{max})$ arithmetic operations (for both the unconstrained and the two constrained versions of our problem), which is super-polynomial in the input length (being linear in s_{max}). Our approximation algorithms can find c-approximate, for $c \sim 2$, solutions with $O(n)$ arithmetic operations (for both the unconstrained and the two constrained versions of our problem), and a $(1+\epsilon)$-approximate solution with $O(n * \text{polylog}(s_{max}))$ arithmetic operations (for both the unconstrained and one constrained version of our problem). The latter algorithm can be shown to maintain a $(1+\epsilon)$-approximated solution by only requiring $O(1)$ amortized arithmetic operations over a sequence of record additions and deletions, for any $\epsilon > 0$ and under very general parameter settings. (Descriptions of this result and our formal privacy model, and almost all proofs are omitted due to space restrictions.)

Related work. In the model of databases outsourced to cloud servers, there is a significant amount of work on data encryption (seemingly originated in [7]), and some amount of work on privacy-preserving database retrieval protocols (see, e.g., [1]), and query-based access policy compliance (see, e.g. [2]). Minimizing the record length is an old problem considered in contexts like statistics and memory management, and that does not appear to have been investigated with an algorithmic viewpoint. An unconstrained version of our problem was introduced by [10], who showed that when record lengths follow some classes of continuous probability distributions, the optimal choice of target record length is a quantity close to our result for arbitrary distributions. In [9], the author analyzed this problem in terms of the characteristic function of the distribution of the record length, and gave solutions for the cases of the uniform, exponential, and geometric distributions. In [4] and [3], the authors considered several different target record sizes, and presented solutions based on dynamic programming and non-linear optimization techniques. In [8], the author considers a similar problem in an extended model (somehow merging multiple records into one), which does not preserve some of the database search functionalities. None of these works focused on minimizing the running time required to produce a solution for an arbitrary distribution of a discrete set of record lengths. We are also not aware of any work considering constrained versions of this problem or the case of dynamic databases.

2 Definitions, Privacy and Algorithmic Models

Preliminary definitions. A *database* is an indexed sequence of records that can be modified, added, and removed over time. We denote as n the current number of records in the database. For $i \in \{1, \ldots, n\}$, the i-th database record can be seen as composed of two parts: the *header*, whose size $a \geq 1$ is constant across all records, and the *payload*, whose size $s_i \geq 1$ is variable. We denote as s_{max} the maximum integer among $s_1, \ldots, s_n$. We define a *fixed record length database* as a database where all record lengths are equal.

Privacy model. We consider the following *Private Database Outsourcing* (PDO) problem: a data owner, on input a database, wants to outsource some version of the database to a cloud server so that the server only learns minimal information about the database content, and yet can engage in database retrieval protocols with one or more clients, as well as policy compliance protocols ensuring that clients' queries are authorized. In this paper we only deal with the database outsourcing part of the PDO problem, but note that our approach integrates well with privacy-preserving database retrieval solutions (e.g., from [1]) and privacy-preserving policy compliance solutions (e.g., from [2]). Encryption is a natural candidate tool to keep the outsourced database private from the server, who can still later run database retrieval and policy compliance protocols using techniques based on computing over encrypted data. Although in the cryptographic literature leakage of the length of an encrypted plaintext is usually considered a very minimal privacy violation (both the formal definitions of encryption [6] and of 2-party and multi-party private function evaluation protocols [11, 5] admit leakage of plaintext/input lengths), leaking the lengths $s_1, \ldots, s_n$ of all database records may well be an unacceptable privacy loss. We then ask the natural privacy question of what could/should be kept private in any solution to the PDO problem. In a similar question on the encryption of multiple different-length messages, solutions used in practice include either (a) padding each message to its next block length, which leaks a close upper bound of all length values; or (b) padding all messages to a common block length, which, although not the most efficient solution, has more satisfactory privacy as it only leaks an upper bound of all length values. Both (a) and (b) leak the exact number of encrypted messages.

(Informal) Privacy requirement: In formulating our privacy model, we attempt to capture the satisfactory privacy properties of the solution approach in (b), and at the same time generalize it to allow for a richer set of solutions to the PDO problem. Specifically, we require any solution to the PDO problem to leak at most:

1. the output of a function that is *symmetric*[1] over the record lengths $s_1, \ldots, s_n$;
2. an upper bound on the number n of database records.

A formal description of this requirement can be provided in the simulation-based privacy framework and is omitted due to space restrictions.

Algorithmic Model. To transform any database into a fixed record length one, we only use two types of operations: (1) *padding* a payload; i.e., concatenating the payload with a predefined string (e.g., a 1 followed by an all 0's string), and (2) *σ-splitting* a record

[1] A function f is *symmetric* if it satisfies $f(x_1, \ldots, x_n) = f(x_{\rho(1)}, \ldots, x_{\rho(n)})$ for any input $(x_1, \ldots, x_n)$ and any permutation ρ over $\{1, \ldots, n\}$.

into multiple smaller records, where each new record has a copy of the same a-bit header and a distinct σ-bit piece of the original record's payload, where the last record may be padded so to have a σ-bit payload as well. Note that while the total length of the i-th record is $a + s_i$ in the original database, this length becomes $a + \sigma$ in the fixed record length database, for some σ that can be chosen from a set of allowed values P. In particular, we can consider, without loss of generality, $P = \{1, \ldots, s_{max}\}$, in which case for *any* $\sigma \in P$, any above defined database can be transformed into a fixed record length database via the following sequence of padding and σ-splitting operations, as follows: each record with payload shorter than σ can be padded and each s_i-bit record with payload longer than σ can be split into $\lceil s_i/\sigma \rceil$ records, the last one being padded. Then, we can compute the size of the fixed record length database as function

$$f(\sigma) = \sum_{i \in [n]} \left\lceil \frac{s_i}{\sigma} \right\rceil (\sigma + a), \tag{1}$$

where the input σ is taken from the set P which will usually be $\{1, \ldots, s_{max}\}$ or a subset of that, although in our analysis we will often abuse notation to consider σ taken from the set of real numbers. We then define our problem of interest as the problem of minimizing the Encrypted Database Size (EDS), as captured by the function f defined in Formula 1, and coming into two main variants, depending on the constraints that we pose on the number of splitting operations. In the first variant, denoted as maxEDS, there is a maximum number of pieces into which we may split any record payload. In the second variant, denoted as avgEDS, there is a maximum total number of pieces into which we may split all of the record payloads. Formal definition follows.

Definition 1. [EDS] Given $n+1$ positive integers a, $s_1, \ldots, s_n$, find the integer σ from set $P = \{1, \ldots, s_{max}\}$ that minimizes $f(\sigma)$.

Definition 2. [maxEDS] Given $n + 2$ positive integers a, $s_1, \ldots, s_n, c_{max}$, find the integer σ from set $P = \{1, \ldots, s_{max}\}$ that minimizes $f(\sigma)$ subject to the constraints $\lceil s_i/\sigma \rceil \leq c_{max}$, for $i = 1, \ldots, n$.

Definition 3. [avgEDS] Given $n + 2$ positive integers a, $s_1, \ldots, s_n$ and c_{avg}, find the integer σ from set $P = \{1, \ldots, s_{max}\}$ that minimizes $f(\sigma)$ subject to the constraint $\sum_{i=1}^{n} \lceil s_i/\sigma \rceil \leq c_{avg}$.

We investigate algorithms that solve these problems either exactly (i.e., returning any σ^* that minimizes $f(\sigma)$), or δ-approximately (i.e., returning a σ such that $f(\sigma) \leq \delta f(\sigma^*)$), or δ-approximately even across a sequence of record additions and deletions. We will not try the approach of naturally extending f so that it is defined over all real numbers, and finding an analytical expression for a σ that exactly minimizes f as f is not convex, and is discontinuous at many points. To meet our privacy requirement, we will only design algorithms $\mathcal{A}$ which return a value σ that is a symmetric function of the values $s_1, \ldots, s_n$. For the exact algorithms, this can be easily verified since these algorithms return the value σ that minimizes the function f defined in Formula 1, and f is a symmetric function of $s_1, \ldots, s_n$. For the approximate algorithms, this is verified by direct inspection that the formula used in each of these algorithms is also a symmetric function of $s_1, \ldots, s_n$.

3 Exact algorithms

We discuss two exact algorithms: a naive algorithm that runs in time $O(n\sigma_{max})$ and an improved algorithm that runs in time $O(n + \sigma_{max} \log \sigma_{max})$.

A first exact algorithm. On input $a, s_1, \ldots, s_n$, define algorithm $\mathcal{A}_{0,1}$ for the EDS problem as follows: for each value σ from P, evaluate $f(\sigma)$ using Formula (1); finally, select the value σ that minimizes $f(\sigma)$. This algorithm finds an optimal solution for EDS in $O(n|P|) = O(ns_{max})$ arithmetic operations. By further checking that the constraint is satisfied, this algorithm is directly extended to find an optimal solution for maxEDS and avgEDS in the same asymptotic running time.

A second exact algorithm. As a potential improvement, we consider algorithm $\mathcal{A}_{0,2}$ that precomputes two sets of values related to the multiplicities of the payload sizes, and uses an alternative expression for $f(\sigma)$ that is faster to compute, given the precomputed values. Specifically, the precomputed values consist of the multiplicities of the payload sizes $m_v = |\{i \in N : s_i = v\}|$, and the related values $m_v^+ = |\{i \in N : s_i \geq v\}|$, defined for all $v \in P$. We note the following

Fact 1. For each $v = 1, \ldots, s_{max}$, it holds that $m_v^+ = m_{v+1}^+ + m_v$.

The alternative expression for $f(\sigma)$ is the following:

$$f(\sigma) = \sum_{h=1}^{\lceil s_{max}/\sigma \rceil} \sum_{i:\lceil s_i/\sigma \rceil = h} \left\lceil \frac{s_i}{\sigma} \right\rceil (\sigma + a) \tag{2}$$

$$= \sum_{h=1}^{\lceil s_{max}/\sigma \rceil} h \left(m^+_{1+\sigma(h-1)} - m^+_{1+\sigma h} \right) (\sigma + a) \tag{3}$$

Based on these definitions, we define algorithm $\mathcal{A}_{0,2}$ for the EDS problem, as follows:

Algorithm $\mathcal{A}_{0,2}$: On input $a, s_1, \ldots, s_n$, do the following:

1. Calculate s_{max}
2. Calculate m_v, for each v from 1 to s_{max}.
3. Calculate m_v^+, for each v from s_{max} down to 1, using recurrence relation in Fact 1.
4. Calculate $f(\sigma)$, for each σ in P, using Formula (3).
5. Return the value σ_f^* that minimizes f.

For the running time, we observe that steps 1, 2 and 3 can be run in time $O(n)$, step 4 in time $O(s_{max} \log s_{max})$ (that is, $O(s_{max}/\sigma)$, for $\sigma = 1, \ldots, s_{max}$), and step 5 in time $O(s_{max})$. This algorithm is directly extended to find an optimal solution for maxEDS by further checking that the constraint is satisfied, which can be done in time $O(n)$. Checking that the constraint for avgEDS is satisfied can be done in time $O(s_{max} \log s_{max})$ by observing that the left hand side of the constraint can be rewritten similarly as in Formula (3). Thus, algorithm $\mathcal{A}_{0,2}$ can be extended to work for maxEDS and avgEDS by keeping the same asymptotic running time. We obtain the following

Theorem 1. For each of the problems EDS, maxEDS and avgEDS, we can construct an algorithm that exactly solves the problem in $O(\min\{ns_{max}, n + s_{max} \log s_{max}\})$ arithmetic operations.

The running time in Theorem 1 is not polynomial in the input length (as it is linear in s_{max}) and can be too expensive in practical large databases (e.g., when $n \geq 10^9$ and $s_{max} \geq 10$MB). Thus, we turn our attention to finding approximation algorithms with faster running times (possibly, linear in n and polylogarithmic in s_{max}). As both the algorithm's running time and the quality of the approximation are of interest, we study algorithms that attempt to minimize one metric while achieving satisfactory performance on the other one. Specifically, we study algorithms with constant approximation factor and very fast running time (i.e., $O(n)$) in Section 4, and algorithms with very small (i.e., $(1+\epsilon)$, for any $\epsilon > 0$) approximation factor and any running time improving over $\mathcal{A}_{0,1}$ and $\mathcal{A}_{0,2}$ in Section 5.

4 Faster Algorithms with O(1) Approximation Factor

In this section we study algorithms for maxEDS and avgEDS that attempt to minimize their running time while achieving a constant approximation factor. When compared with the exact algorithms in Section 3, the algorithms in this section achieve a smaller running time (only linear in n and with no dependency on s_{max}) at the cost of achieving approximation factor 2 or slightly greater than 2. We start with a definition useful for both algorithms; we define function g over the set of real numbers, as follows:

$$g(\sigma) = \sum_{i=1}^{n} \left(\frac{s_i}{\sigma} + 1 \right) (\sigma + a), \tag{4}$$

Then, our first algorithm, for problem maxEDS, and its properties are as follows.

Algorithm $\mathcal{A}_1^{max}$: On input $a, s_1, \ldots, s_n, c_{max}$, compute $\bar{s} = \sum_{i=1}^{n} s_i/n$ and then $\sigma_g^* = \sqrt{a\bar{s}}$, and then return σ, computed as the value among $\lfloor \sigma_g^* \rfloor$, $\lceil \sigma_g^* \rceil$, $\lceil s_{max}/c_{max} \rceil$ that is at least $\lceil s_{max}/c_{max} \rceil$ and results in the smaller value for $f(\sigma)$.

Theorem 2. $\mathcal{A}_1^{max}$ 2-approximately solves the maxEDS problem in $O(n)$ arithmetic operations.

An algorithm $\mathcal{A}_1$ for problem EDS with the same running time and approximation factor can be obtained by directly simplifying $\mathcal{A}_1^{max}$. We cannot directly adapt these techniques to avgEDS, as a value for σ satisfying the equality in constraint $\sum_{i=1}^{n} \lceil s_i/\sigma \rceil \leq c_{avg}$ may be hard to find, due to the n rounding operations. Instead, we minimize the approximating function g subject to an even tighter constraint, and bound the additional error produced. We now define an algorithm for avgEDS and state its properties.

Algorithm $\mathcal{A}_1^{avg}$: On input $a, s_1, \ldots, s_n, c_{avg}$, compute the quantities $\beta = c_{avg}/n$ and $\bar{s} = \sum_{i=1}^{n} s_i/n$. Then check whether $\left\lfloor \sqrt{a\bar{s}} \right\rfloor \geq \bar{s}/(\beta - 1)$. If yes, set s_g^* the value among $\left\lfloor \sqrt{a\bar{s}} \right\rfloor$ and $\left\lceil \sqrt{a\bar{s}} \right\rceil$ that has a lower value of function g, as defined in Formula 4. If not, set $s_g^* = \lceil \bar{s}/(\beta - 1) \rceil$. Finally, return s_g^*.

Theorem 3. $\mathcal{A}_1^{avg}$ $(2+1/\bar{s}+1/(\beta(\beta-1)))$-approximately solves the avgEDS problem in $O(n)$ arithmetic operations, where $\beta = c_{avg}/n$.

5 Fast Algorithms with $(1+\epsilon)$ Approximation Factor

We show an algorithm for maxEDS (and thus EDS) that achieves $(1+\epsilon)$ approximation factor, for any $\epsilon > 0$, and running time asymptotically smaller than the exact algorithms described in Section 4. Our algorithm extends the technique used in Section 4 where we approximated function f by a convex function g. Here, we approximate function f with several functions g_k, for any integer $k > 0$. First, for any integers $k > 0$ and any real number $x > 0$, we define $w(k, x)$ as $\lceil x \rceil$ if $x \leq k-1$ or $1+x$ if $x > k-1$. Then, for any integer $k > 0$, we define an approximation function g_k to f as

$$g_k(\sigma) = \sum_{i \in N} w\left(k, \frac{s_i}{\sigma}\right)(\sigma + a)$$

Note that for $k = 1$, function g_1 is the same as function g defined in Section 4, where we have showed that g can be computed in time $O(n)$. For $k > 1$, we can still evaluate function g_k more efficiently than f, using some auxiliary variables and a suitable piece-wise decomposition and rewriting of g_k. A first set of definitions relevant to this goal include the previously defined quantities m_v, m_v^+ and some new quantities $b_v, c_{h,v}$, defined as follows, for each $v = 1, \ldots, s_{max}$, and each integer $h > 0$:

1. $b_v = \sum_{s_i \geq v} s_i$; and
2. $c_{h,v} = |\{i \in N : \lceil s_i/v \rceil = h\}|$.

To reduce the number of candidate σ values from P from which we plan to evaluate $g_k(\sigma)$, for any integer k and any $j = 0, \ldots, q$, where q is the max value such that $(1+\epsilon)^q * \lceil s_{max}/c_{max} \rceil \leq s_{max}$, we define the quantities:

1. $\tau_0 = \lceil s_{max}/c_{max} \rceil$, $\tau_j = \lceil (1+\epsilon) * \tau_{j-1} \rceil$;
2. $M_j = |\{i \in N : 1 + \tau_j(k-1) \leq s_i < \tau_{j+1}(k-1)\}|$; and
3. $B_j = \sum_{s_i \geq 1+\tau_j(k-1)}^{\tau_{j+1}(k-1)} s_i$.

By applying the definitions of $m_v, M_j, b_v, B_j, c_{h,\sigma}, m_v^+$, we derive the following

Fact 3. For each $j = 0, \ldots, q$ and each integer $h > 0$, it holds that

1. $m^+_{1+\tau_j(k-1)} = m^+_{\tau_{j+1}(k-1)} + M_j$
2. $b_{1+\tau_j(k-1)} = b^+_{\tau_{j+1}(k-1)} + B_j$
3. $c_{h,\tau_j} = m^+_{1+\tau_j(h-1)} - m^+_{1+\tau_j h}$

We can then evaluate $g_k(\sigma)$, for all $\sigma = \tau_0, \tau_1, \ldots, \tau_q$, as

$$g_k(\sigma) = \sum_{h=1}^{k-1} \sum_{i:\lceil s_i/\sigma \rceil = h} \left\lceil \frac{s_i}{\sigma} \right\rceil (\sigma + a) + \sum_{i: s_i/\sigma > k-1} \left(1 + \frac{s_i}{\sigma}\right)(\sigma + a) \tag{5}$$

$$= \left(\sum_{h=1}^{k-1} c_{h,\sigma} h\right)(\sigma + a) + \left(m^+_{1+\sigma(k-1)} + b_{1+\sigma(k-1)} \frac{1}{\sigma}\right)(\sigma + a) \tag{6}$$

Based on the above, we can now construct our algorithm $\mathcal{A}_2^{max}$, by setting $k = \lceil 1/\epsilon \rceil$, quickly computing g_k based on Formula 6 and Fact 3, approximating f with g_k, and minimizing g_k subject to the constraint $\lceil s_{max}/\sigma \rceil \leq c_{max}$ and σ varying over the multiplicative grid of $\tau_0, \ldots, \tau_q$.

Algorithm $\mathcal{A}_2^{max}$: On input $a, s_1, \ldots, s_n, c_{max}, \epsilon$, do the following:

1. Calculate s_{max} and set $k = \lceil 1/\epsilon \rceil$.
2. Calculate all M_j, for $j = 0, \ldots, q$ by scanning all s_i's only once and, for each s_i, using binary search to find the associated M_j.
3. Calculate all B_j, for $j = 0, \ldots, q$ by scanning all s_i's only once and, for each s_i, using binary search to find the associated B_j.
4. Calculate $m^+_{1+\tau_j(k-1)}$, for $j = 0, \ldots, q$, using recurrence relation in Fact 3, item 1.
5. Calculate $b_{1+\tau_j(k-1)}$, for $j = 0, \ldots, q$, using recurrence relation in Fact 3, item 2.
6. Calculate c_{h,τ_j}, for $j = 0, \ldots, q$, and $h = 1, \ldots, k-1$, using recurrence relation in Fact 3, item 3.
7. Choose the value $\sigma^*_{g,k}$ that minimizes g_k, by examining every value of σ from $\{\tau_0, \tau_1, \ldots, \tau_q\}$ and using Formula (6) to evaluate $g_k(\sigma)$.

Theorem 4. For any $\epsilon > 0$, $\mathcal{A}_2^{max}$ $(1+\epsilon)$-approximately solves the maxEDS problem in $O(n \log(\log(c_{max})/\epsilon) + \log c_{max}/\epsilon^2)$ arithmetic operations.

Acknowledgement. We thank Euthimios Panagos for interesting discussions. This work was supported by the Intelligence Advanced Research Projects Activity (IARPA). The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation hereon. Disclaimer: The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of IARPA, DoI/NBC, or the U.S. Government.

References

1. G. Di Crescenzo, D. Cook, A. McIntosh, E. Panagos: "Practical Private Information Retrieval from a Time-Varying, Multi-Attribute, and Multiple-Occurrence Database". In *Proc. of 28th DBSec Conference*, 2014.
2. G. Di Crescenzo, J. Feigenbaum, D. Gupta, E. Panagos, J. Perry, R. Wright: "Practical and Privacy-Preserving Policy Compliance for Outsourced Data". In *Proc. of 2nd WAHC Workshop*, 2014.
3. R. E. Erickson, S. Halfin and H. Luss, "Optimal Sizing of Records when Divided Messages Can Be Stored in Records of Different Sizes", *Operations Research*, vol. **30**, pp. 29-39, 1982
4. R. E. Erickson and H. Luss, "Optimal Sizing of records Used to Store Messages of Various Lengths", *Management Science*, vol. **26**, pp. 796-809, 1980
5. O. Goldreich, S. Micali, and A. Wigderson, "How to Play any Mental Game or A Completeness Theorem for Protocols with Honest Majority", *Proc. of ACM STOC 1987*: 218-229.
6. S. Goldwasser, and S. Micali, "Probabilistic Encryption", *J. Comput. Syst. Sci.*, vol. 28(2): 270-299 (1984)
7. H. Hacigms, B. R. Iyer, C. Li, S. Mehrotra: "Executing SQL over encrypted data in the database-service-provider model". In *Proc. of SIGMOD Conference* 2002, pp. 216-227
8. H. Luss, "An Extended Model for the Optimal Sizing of Records", *Journal of Operation Research Society*, vol. **34**, pp. 1099-1105, 1983
9. P. Sipala, "Optimum Cell Size for the Storage of Messages" *IEEE Transactions on Software Engineering*, vol. **SE-7**, pp. 132-134, 1981
10. E. Wolman, "A Fixed Optimum Cell-Size for Records of Various Lengths", *Journal of the ACM*, vol. **12**, pp. 53-70, 1965
11. A. C. Yao, "How to Generate and Exchange Secrets", *Proc. of IEEE FOCS 1986*: 162-167