IJCSE International Journal of Computer Sciences and Engineering Open Access
Survey Paper Vol.-7, Special Issue, 4, Feb 2019 E-ISSN: 2347-2693

A Trusted Hardware- Database Based with Data Confidentiality

S. Bavithra[1*], T. Manivannan[2]

[1,2]Dept. of Computer Science , E.G.S.Pillay Arts and Science College, Nagapattinam, Tamilnadu, India.

Corresponding Author: bavithrakkl1995@gmail.com

Available online at: www.ijcseonline.org

***Abstract*—**Traditionally, as soon as confidentiality becomes a concern, data are encrypted before outsourcing to a service provider. Any software-based cryptographic constructs then deployed, for server-side query processing on the encrypted data, inherently limit query expressiveness. Here, we introduce TrustedDB, an outsourced database prototype that allows clients to execute SQL queries with privacy and under regulatory compliance constraints by leveraging server-hosted, tamper-proof trusted hardware in critical query processing stages, thereby removing any limitations on the type of supported queries. Despite the cost overhead and performance limitations of trusted hardware, we show that the costs per query are orders of magnitude lower than any (existing or) potential future software-only mechanisms. TrustedDB is built and runs on actual hardware, and its performance and costs are evaluated here.

***Keywords*—**Encryption,SQL,Query,DB.

I. INTRODUCTION

Although the benefits of outsourcing and clouds are well known [41], significant challenges yet lie in the path of large-scale adoption since such services often require their customers to inherently trust the provider with full access to the outsourced data sets. Numerous instances of illicit insider behavior or data leaks have left clients reluctant to place sensitive data under the control of a remote, third-party provider, without practical assurances of privacy and confidentiality, especially in business, health-care, and government frameworks. Moreover, today’s privacy guarantees for such services are at best declarative and subject customers to unreasonable fine-print clauses. For example, allowing the server operator to use customer behavior and content for commercial profiling or govern-mental surveillance purposes.

Existing research addresses several such security as-pects, including access privacy and searches on encrypted data. In most of these efforts, data are encrypted before outsourcing. Once encrypted however, inherent limitations in the types of primitive operations that can be performed on encrypted data lead to fundamental expressiveness and practicality constraints.

Recent theoretical cryptography results provide hope by proving the existence of universal homomorphisms, i.e., encryption mechanisms that allow computation of arbitrary functions without decrypting the inputs. Unfortunately, actual instances of such mechanisms seem to be decades away from being practical [17].

Ideas have also been proposed to leverage tamper-proof hardware to privately process data server-side, ranging from smart-card deployment in healthcare to more general database operations.

Yet, common wisdom so far has been that trusted hardware is generally impractical due to its performance limitations and higher acquisition costs. As a result, with very few exceptions, these efforts have stopped short of proposing or building full-fledged database processing engines.

However, recent insights [9] into the cost-performance tradeoff seem to suggest that things stand somewhat differently. Specifically, at scale, in outsourced contexts, computation inside secure processors is orders of magnitude cheaper than any equivalent cryptographic operation per-formed on the provider’s unsecured server hardware, despite the overall greater acquisition cost of secure hardware.

This is so because the overheads for cryptography that allows some processing by the server on encrypted data are extremely high even for simple operations. This fact is rooted not in cipher implementation inefficiencies but rather in fundamental cryptographic hardness assumptions and constructs, such as trapdoor functions. Moreover, this is unlikely to change anytime soon as none of the current primitives have, in the past half-century. New mathematical hardness problems will need to be discovered to allow hope of more efficient cryptography.

As a result, we posit that a full-fledged privacy enabling secure database leveraging server-side trusted hardware can be built and run at a fraction of the cost of any (existing or future) cryptography-enabled private data processing on common server hardware. We validate this by designing and building TrustedDB, an SQL database processing engine that makes use of tamper-proof cryptographic coprocessors such as the IBM 4764 [3] in close proximity to the outsourced data.

Tamper resistant designs, however, are significantly constrained in both computational ability and memory capacity which makes implementing fully featured data-base solutions using secure coprocessors (SCPUs) very challenging. TrustedDB achieves this by utilizing common unsecured server resources to the maximum extent possible. For example, TrustedDB enables the SCPU to transparently access external storage while preservingdata confidentiality with on-the-fly encryption. This elim-inates the limitations on the size of databases that can be supported. Moreover, client queries are preprocessed to identify sensitive components to be run inside the SCPU. Nonsensitive operations are off-loaded to the untrusted host server. This greatly improves performance and reduces the cost of transactions.

Overall, despite the overheads and performance limita-tions of trusted hardware, the costs of running TrustedDB are orders of magnitude lower than any (existing or) potential future cryptography-only mechanisms. Moreover, it does not limit query expressiveness.

The contributions of this paper are threefold: 1) the introduction of new cost models and insights that explain and quantify the advantages of deploying trusted hardware for data processing; 2) the design, development, and evaluation of TrustedDB, a trusted hardware based rela-tional database with full data confidentiality; and 3) detailed query optimization techniques in a trusted hardware-based query execution model.

II. THE REAL COSTS OF SECURITY

As soon as confidentiality becomes a concern, data need to be encrypted before outsourcing. Once encrypted, solutions can be envisioned that: (A) straightforwardly transfer data back to the client where it can be decrypted and queried,deploy cryptographic constructs server-side to process encrypted data, and (C) process encrypted data server-side inside tamper-proof enclosures of trusted hardware.

In this section, we will compare the per-transaction costs of each of these cases. This is possible in view of novel results of Chen and Sion [9] that allow such quantification. We will show that, at scale, in outsourced contexts,computation inside secure hardware processors is orders of magnitude cheaper than any equivalent crypto-graphic operation performed on the provider's unsecured common server hardware (B). Moreover, due to the extremely high cost of networking as compared with computation, the overhead of transferring even a small subset of the data back to the client for decryption and processing in (A) is overall significantly more expensive than (C).

The main intuition behind this has to do with the amortized cost of CPU cycles in both trusted and common hardware, as well as the cost of data transfer. Due to economies of scale, provider-hosted CPU cycles are 1-2 orders of magnitude cheaper than that of clients and oftrusted hardware. The cost of a CPU cycle in trusted hardware (56+ picocents,1 discussed below) becomes thus of the same order as the cost of a traditional client CPU cycle at (e.g., 14-27 picocents for small businesses) including acquisition and operating costs.
Additionally, when data are hosted far from their accessing clients, the extremely expensive network traffic often dominates. For example, transferring a single bit of data over a network costs upwards of 3,500 picocents [9].Finally, cryptography that would allow processing on encrypted data demands extremely large numbers of cycles even for very simple operations such as addition.

Cost of Primitives
Compute cycles and networks. In [9], Chen and Sion derived the cost of compute cycles for a set of environments ranging from individual homes with a few PCs (H) to large enterprises and compute clouds (L) (M, L ¼ medium-, large-sized business). These costs include a number of factors, such as hardware (server, networking), building (floor space leasing), energy (electricity), service (personnel, maintenance), and so on. Their main thesis is that, due to economies of scale and favorable operating parameters, per-cycle costs decrease dramatically when run in large compute providers' infrastructures.

The resulting CPU cycle costs (see Fig. 1) range from 27 picocents for a small business environment to less than half of a picocent for large cloud providers. Network service costs range from a few hundred picocents per bit for nondedicated service to thousands of picocents in the case of medium-sized businesses. Detailed numbers are available in [9].

Also, the work in [39], [9] derives the cost of x86-equivalent CPU cycles inside cloud-hosted SCPUs such as the IBM 4764 to be 56 picocents. We note that while this is indeed much higher than the <0:5 picocent cost of a cycle on commodity hardware, it is comparable to the cost of cycles in CPUs hosted in small-sized enterprises (14-27 picocents).

Comparison

Given these data points, we now compare the A, B, and C alternatives discussed above. We consider the following simple scenario. A client outsources an encrypted data set composed of integers to a provider. The encrypted data are then subjected to a simple aggregation (SUM) query in which the server is to add all the integers without decryption and return the result to the client. We chose this mechanism not only for its illustrative simplicity but also because SUM aggregation is one of the very few types of queries for which nonhardware solutions have been proposed. This allows us to directly compare with existing work. Later in Section 2.3, we also generalize for arbitrary queries. Fig. 2 summarizes the cost analysis that follows.

Querying unencrypted data. No confidentiality. As a base-line, consider the most prevalent scenario today, in which the client's data are stored unencrypted with the service provider. Client queries are executed entirely on the provider's side and only the results are transferred back. Although this is the most cost-effective solution, it offers no

Generalized Argument

Recall that current cryptographic constructs are based on trapdoor functions [18]. Currently, viable trapdoors are based on modular exponentiation in large fields (e.g., 2,048 bit modular operations) and viable homomorphisms involve a trapdoor for computing the ciphertexts. Addition-ally, the homomorphic operation itself involves processing these encrypted values at the server in large fields, while respecting the underlying encryption trapdoor, incurring at least the cost of a modular multiplication [33], [28], [29]. This fundamental cryptography has not improved in efficiency in decades and would require the invention of new mathema-tical tools before such improvements are possible.
Thus, overall, for large-scale, efficient deployments (e.g., clouds) where CPU cycles are extremely cheap (e.g., 0.45 picocents/cycle), performing the cheapest, least secure homomorphic operations (modular multiplication) comes at a price tag of at least 30,000 picocents [9] even for values as small as 32-bit (e.g., salaries and ZIP codes).

Thus, even if we assume that in future developments homomorphisms will be invented that can allow full Turing Machine languages to be run under the encryption envelope, unless new trapdoor math is discovered, each operation will yet cost at least 30,000 picocents when run on efficient servers. By comparison, SCPUs process data at a cost of 56 picocents/cycle. This is a difference of several orders of magnitude in cost. We also note that, while ECC signatures (e.g., even the weak ECC-192) may be faster, ECC-based trapdoors would be even more expensive, as they would require two point multiplications, coming at a price tag of least 780,000 cycles (see [11, p. 402]).

Yet, this is not entirely accurate, as we also need to account for the fact that SCPUs need to read data in before processing. The SCPUs considered here feature a decryption throughput of about 10-14 MB/second for AES decryption [3], confirmed also by our benchmarks. This limits the ability to process data. For example, comparing two 32-bit integers as in a JOIN operation becomes dominated not by the single-cycle conditional JUMP CPU operation but by the cost of decryption. At 166-200 megacycles/second, this results in the SCPU having to idly wait anywhere between 47 and 80 cycles for decryption to happen in the crypto engine module before it can process the data. This in effect results in an amortized SCPU cost of between 2,632 and 4,480 picocents (3,556 picocents on average) for each operation which reduces the above three orders of magnitude difference to only one order of magnitude, still in favor of SCPUs.4

The above holds even for the case when the SCPU has only enough memory for the two compared values. Further, in the presence of significantly higher, realistic amounts of SCPU memory (e.g., M ¼ 32 MB for 4764-001), optimiza-tions can be achieved for certain types of queries such as relational JOINs. The SCPU can read in and decrypt entire data pages instead of single data items and run the JOIN query over as many of the decrypted data pages as would fit in memory at one time. This results in significant savings. To illustrate, consider a page size of P 32-bit words and a simple JOIN algorithm for two tables of size N 32-bit integers each (we are just concerned with the join attribute). Then, the SCPU will perform a number of ðN=P Þ2 þ ðN=P Þ page fetches each involving also a page data decryption at a cost of P 3,556 picocents. Thus, we get a total cost of ðNP2 þ NÞ 3;556 þ N2 56. For reasonable sizes, e.g., P ¼ M=2=4 ¼ 4 ~~mil~~lion, this cost becomes 3þ orders of magnitude lower than the N2 30;000picocent cost incurred in the cryptography-based case.

Cost versus Performance. Given these 3þ orders of magnitude cost advantages of the SCPU over cryptogra-phy-based mechanisms, we expect that for the above discussed aggregation query mechanism [42], the SCPU's overall performance will also be at least comparable if not better despite the CPU speed handicap. We experimentally evaluated this hypothesis and achieved a throughput of about 1.07 million tuples/second for the SCPU. By contrast, in [42], best case scenario throughputs range between 0.58 and 0.92 million tuples/second and at much higher overall cost.

III. ARCHITECTURE

To overcome SCPU storage limitations, the outsourced data are stored at the host provider's site. Query processing engines are run on both the server and in the SCPU. Attributes in the database are classified as being either public or private. Private attributes are encrypted and can only be decrypted by the client or by the SCPU.

Since the entire database resides outside the SCPU, its size is not bound by SCPU memory limitations. Pages that need to be accessed by the SCPU-side query processing are pulled in on demand by the Paging Module.
Query execution entails a set of stages:

1. In the first stage, a client defines a database schema and partially populates it. Sensitive attributes are marked using the SENSITIVE keyword which the client layer transparently processes by encrypting the corresponding attributes:

   CREATE TABLE customer(ID integer primary key,

   Name char(72) SENSITIVE, Address char(120) SENSITIVE).
2. Later, a client sends a query request to the host server through a standard SQL interface. The query is transparently encrypted at the client site using the public key of the SCPU. The host server thus cannot decrypt the query.

3. The host server forwards the encrypted query to the Request Handler inside the SCPU.

4. The Request Handler decrypts the query and forwards it to the Query Parser. The query is parsed generating a set of plans. Each plan is constructed by rewriting the original client query into a set of subqueries, and, according to their target data set classification, each subquery in the plan is identified as being either public or private.

   The Query Optimizer then estimates the execution costs of each of the plans and selects the best plan (one with least cost) for execution forwarding it to the dispatcher.

6. The Query Dispatcher forwards the public queries to the host server and the private queries to the SCPU database engine while handling dependencies. The net result is that the maximum possible work is run on the host server's cheap cycles.

7. The final query result is assembled, encrypted, digitally signed by the SCPU Query Dispatcher, and sent to the client.

Query parsing and execution.Sensitive attributes can occuranywhere within a query, e.g., in SELECT, WHERE, or GROUP-BY clauses, in aggregation operators, or within subqueries. The Query Parser's job is then:

1. To ensure that any processing involving private attributes is done within the SCPU. All private attributes are encrypted using a shared data encryption keys between the client and the SCPU; hence, the host server cannot decipher these attributes (see Section 5).

2. To optimize the rewrite of the client query such that most of the work is performed on the host server.

To exemplify how public and private queries are generated from the original client query, we use examples from the TPC-H benchmark [2]. TPC-H does not specify any classification of attributes based on security. Therefore, we define a attribute set classification into private (encrypted) and public (nonencrypted). In brief, all attributes that convey identifying information about customers, suppliers, and parts are considered private. The resulting query plans, including rewrites into main CPU and SCPU components for TPC-H queries Q3 and Q6 are illustrated in Fig. 4.

For queries that have WHERE clause conditions on public attributes, the server can first SELECT all the tuples that meet the criteria. The private attributes' queries are then performed inside the SCPU on these intermediate results, to yield the final result.
limitation is rooted in fundamental cryptographic hardness assumptions and constructs, such as cryptographic trapdoors, the cheapest we have so far being at least as expensive as modular multiplication [31], which comes at a price tag of upwards of tens of thousands of picocents per operation [9].The above insights lead to (C) being a significantly more cost-efficient solution than (A) and (B). We now detail.

IV. Conclusion and Future Scope

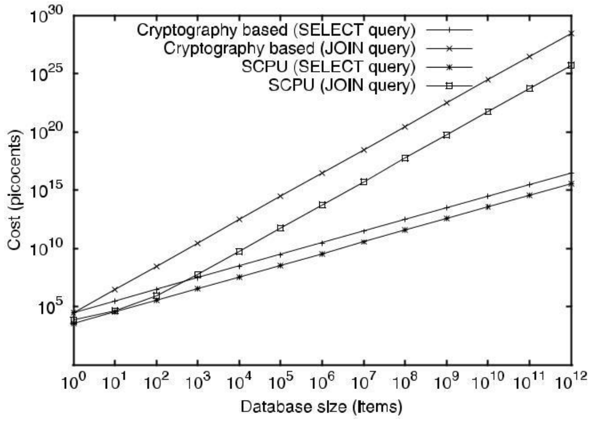

Fig. 3. SCPU is one to three orders of magnitude cheaper than cryptography.

Fig. 3 compares SCPU-based queryprocessing with the most ideal cryptography-based me-chanisms employing a single modular multiplication. Note that such idealistic crypto mechanisms have not been invented yet, but even if they were as Fig. 3 illustrates, for linear processing queries (e.g., SELECT), the SCPU is roughly 1þ order of magnitude cheaper. For JOIN queries, the SCPU costs drop even further even when assuming no available memory. Finally, in the presence of realistic amounts of memory, due to increased overhead amortiza-tion, the SCPU is multiple orders of magnitude cheaper than software-only cryptographic solutions on legacy hardware.

We note that the above conclusion may not apply to targeted niche scenarios. For example, it is entirely possible that by maintaining client precomputed data server-side, processing only a predefined set of queries or by supporting minimal query classes (such as only range queries) specifically designed niche solutions may turn out to be cheaper than general-purpose full-fledged SCPU-backed databases like TrustedDB.

REFERENCES

[1] FIPS PUB 140-2, Security Requirements for Cryptographic Modules, http://csrc.nist.gov/groups/STM/cmvp/standards.html#02, 2013.

[2] TPC-H Benchmark, http://www.tpc.org/tpch/, 2013.

[3] IBM 4764 PCI-X Cryptographic Coprocessor, http://www-03.ibm.com/security/cryptocards/pcixcc/overview.shtml, 2007.

[4] 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," Proc. Conf. Innovative Data Systems Research(CIDR), pp. 186-199, 2005.

[5] A. Iliev and S.W. Smith, "Protecting Client Privacy with Trusted Computing at the Server," IEEE Security and Privacy, vol. 3, no. 2, pp. 20-28, Mar./Apr. 2005.

[6] M. Bellare, "New Proofs for NMAC and HMAC: Security Without Collision-Resistance," Proc. 26th Ann. Int'l Conf. Advances inCryptology, pp. 602-619, 2006.

[7] B. Bhattacharjee, N. Abe, K. Goldman, B. Zadrozny, C. Apte, V.R. Chillakuru, and M. del Carpio, "Using Secure Coprocessors for Privacy Preserving Collaborative Data Mining and Analysis," Proc.Second Int'l Workshop Data Management on New Hardware (DaMoN '06), 2006.

[8] M. Canim, M. Kantarcioglu, B. Hore, and S. Mehrotra, "Building Disclosure Risk Aware Query Optimizers for Relational Data-bases," Proc. VLDB Endowment, vol. 3, nos. 1/2, pp. 13-24, Sept. 2010.

[9] Y. Chen and R. Sion, "To cloud or Not to Cloud?: Musings on Costs and Viability," Proc. Second ACM Symp. Cloud Computing(SOCC '11), pp. 29:1-29:7, 2011.

[10] V. Ciriani, S.D.C. di Vimercati, S. Foresti, S. Jajodia, S. Paraboschi, and P. Samarati, "Combining Fragmentation and Encryption to Protect Privacy in Data Storage," ACM Trans. Information andSystem Security, vol. 13, no. 3, pp. 22:1-22:33, July 2010.

[11] T. Denis, Cryptography for Developers,Syngress, 2007.

[12] E. Damiani, C. Vimercati, S. Jajodia, S. Paraboschi, and P. Samarati, "Balancing Confidentiality and Efficiency in Untrusted Relational DBMSs," Proc. 10th ACM Conf. Computer and Commu-nications Security (CCS '12), 2003.

[13] E. Mykletun and G. Tsudik, "Aggregation Queries in the Database-as-a-Service Model," Proc. 20th IFIP WG 11.3 WorkingConf. Data and Applications Security, pp. 89-103, 2006.

[14] F.N. Afrati and V. Borkar, and M. Carey, and N. Polyzotis, and J.D. Ullman, "Map-Reduce Extensions and Recursive Queries," Proc.14th Int'l Conf. Extending Database Technology (EDBT), pp. 1-8, 2011.

[15] V. Ganapathy, D. Thomas, T. Feder, H. Garcia-Molina, and R. Motwani, "Distributing Data for Secure Database Services," Proc.Fourth Int'l Workshop Privacy and Anonymity in the Information Soc. (PAIS '11), pp. 8:1-8:10, 2011.

[16] T. Ge and S. Zdonik, "Fast Secure Encryption for Indexing in a Column-Oriented DBMS," Proc. IEEE 23rd Int'l Conf. Data Eng.(ICDE), 2007.

[17] R. Gennaro, C. Gentry, and B. Parno, "Non-Interactive Verifiable Computing: Outsourcing Computation to Untrusted Workers," Proc.30th Ann. Conf.Advances in Cryptology (CRYPTO '10), pp. 465-482, 2010.

[18] O. Goldreich, Foundations of Cryptography I. Cambridge Univ. Press, 2001.

[19] B.I.H. Hacigumus and S. Mehrotra, "Efficient Execution of Aggregation Queries over Encrypted Relational Databases," Proc.Ninth Int'l Conf. Database Systems for Advanced Applications, vol. 2973, pp. 633-650, 2004.

[20] H. Hacigumus, B. Iyer, C. Li, and S. Mehrotra, "Executing SQL over Encrypted Data in the Database-Service-Provider Model," Proc. ACM SIGMOD Int'l Conf. Management of Data (SIGMOD '02),216-227, 2002.