TrustedDB: A Trusted Hardware based Outsourced Database Engine

Sumeet Bajaj
Stony Brook Computer Science
Stony Brook, New York, USA
sbajaj@cs.stonybrook.edu

Radu Sion
Stony Brook Computer Science
Stony Brook, New York, USA
sion@cs.stonybrook.edu

ABSTRACT

TrustedDB [11] is an outsourced database prototype that allows clients to execute SQL queries with privacy and under regulatory compliance constraints without having to trust the service provider. TrustedDB achieves this by leveraging server-hosted tamper-proof trusted hardware in critical query processing stages.

TrustedDB does not limit the query expressiveness of supported queries. And, despite the cost overhead and performance limitations of trusted hardware, the costs per query are orders of magnitude lower than any (existing or) potential future software-only mechanisms. In this demo we will showcase TrustedDB in action and discuss its architecture.

1. INTRODUCTION

Virtually all major "cloud" providers today offer a database service of some kind as part of their overall solution. Numerous startups also feature more targeted data management and/or database platforms.

Yet, significant challenges lie in the path of large-scale adoption. Such services often require their customers to inherently trust the provider with full access to the outsourced datasets. But 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, healthcare and government.

Most of the existing research efforts have addressed such outsourcing security aspects by encrypting the data 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 [6]. Unfortunately actual instances of such mechanisms seem to be decades away from being practical [7].

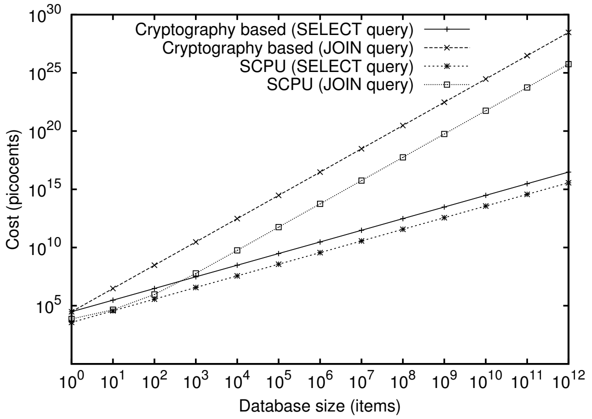


Figure 1: The SCPU is 1-2 orders of magnitude cheaper than deploying cryptography (logarithmic).

TrustedDB on the other hand utilizes secure, tamper resistant hardware such as the IBM 4764/5 [3, 4] cryptographic coprocessors deployed on the service provider's side to implement a complete SQL database processing engine. The TrustedDB design provides strong data confidentiality assurances. Moreover, it does not limit query expressiveness.

2. THE CASE FOR TRUSTED HARDWARE

A cost-based empirical comparison of solutions for query processing using cryptography and trusted hardware [11] (selected results in Figure 1)[1]) shows a 2+ orders of magnitude cost advantage of using trusted hardware over cryptography based mechanisms. This is so because cryptographic overheads (for cryptography that allows some processing by the server) are extremely high even for simple operations, a fact rooted not in cipher implementation inefficiencies but rather in fundamental cryptographic hardness assumptions and constructs (such as trapdoor functions – the cheapest we have so far being at least as expensive as modular multiplication [9]). This is unlikely to change anytime soon (none of the current primitives have, in the past half-century).

Tamper resistant designs provide a secure execution environment for applications, thereby avoiding the need to use expensive cryptographic operations. However, they are significantly constrained in both computational ability and memory capacity which makes implementing fully featured database solutions using secure coprocessors (SCPUs) very

Proceedings of the VLDB Endowment, Vol. 4, No. 12
Copyright 2011 VLDB Endowment 2150-8097/11/08... $ 10.00.

[1] 1 US picocent = 10^{-14} USD

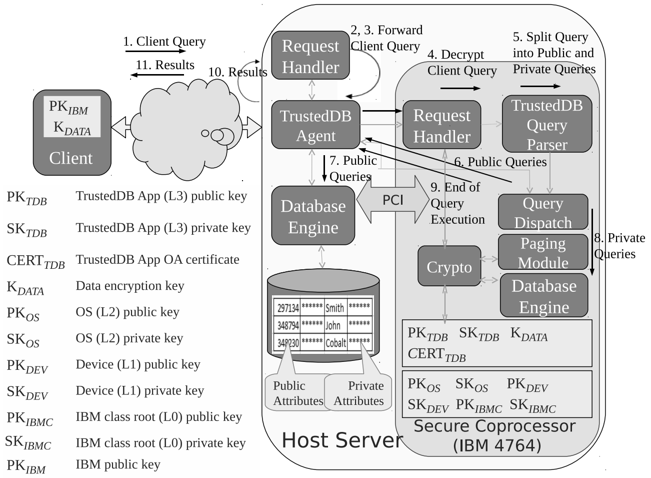

1. Client Query
11. Results
10. Results
2, 3. Forward Client Query
4. Decrypt Client Query
5. Split Query into Public and Private Queries
PK_{IBM}
K_{DATA}
Client
Request Handler
TrustedDB Agent
Request Handler
TrustedDB Query Parser
7. Public Queries
6. Public Queries
Database Engine
PCI
9. End of Query Execution
Query Dispatch
8. Private Queries
Paging Module
Crypto
Database Engine
297134 ****** Smith ******
348794 ****** John ******
348230 ****** Cobalt ******
Public Attributes
Private Attributes
PK_{TDB} SK_{TDB} K_{DATA} $CERT_{TDB}$
PK_{OS} SK_{OS} PK_{DEV} SK_{DEV} PK_{IBMC} SK_{IBMC}
Host Server
Secure Coprocessor (IBM 4764)
PK_{TDB} TrustedDB App (L3) public key
SK_{TDB} TrustedDB App (L3) private key
$CERT_{TDB}$ TrustedDB App OA certificate
K_{DATA} Data encryption key
PK_{OS} OS (L2) public key
SK_{OS} OS (L2) private key
PK_{DEV} Device (L1) public key
SK_{DEV} Device (L1) private key
PK_{IBMC} IBM class root (L0) public key
SK_{IBMC} IBM class root (L0) private key
PK_{IBM} IBM public key

Figure 2: TrustedDB architecture.

challenging. TrustedDB overcomes these limitations by utilizing common unsecured server resources to the maximum extent possible. For example, TrustedDB enables the SCPU to transparently access external storage while preserving data confidentiality with on-the-fly encryption. This eliminates the limitations on the size of databases that can be supported. Moreover, client queries are pre-processed to identify sensitive components to be run inside the SCPU. Non-sensitive operations are off-loaded to the untrusted host server. This greatly improves performance and reduces the cost of transactions.

3. ARCHITECTURE

TrustedDB is built around a set of core components (Figure 2) including a *request handler*, a *processing agent and communication conduit*, a *query parser*, a *paging module*, a *query dispatch module*, a *cryptography library*, and two *database engines*. While presenting a detailed architectural blueprint is not possible in this space, in the following we discuss some of the key elements and challenges faced in designing and building TrustedDB.

3.1 Outline

Challenges. The IBM 4764-001 SCPU presents significant challenges in designing and deploying custom code to be run within its enclosure. For strong security, the underlying hardware code as well as the OS are embedded and no hooks are possible e.g., to augment virtual memory and paging mechanisms. We were faced with the choice of having to provide virtual memory and paging in user land, specifically inside the query processor as well as all the support software. The embedded Linux OS is a Motorola PowerPC 405 port with fully stripped down libraries to the bare minimum required to support the IBM cryptography codebase and nothing else. This constituted a significant hurdle, as cross-compilation became a complex task of mixing native logic with custom-ported functionality. The SCPU communicates with the outside world synchronously through fixed sized messages exchanged over the PCI-X bus in exact sequences. Interfacing such a synchronous channel with the communication model of the query processors and associated paging components required the development of the TrustedDB Paging Module. The SCPU's cryptographic hardware engine features a set of latencies that effectively crippled the ability to run for highly interactive mechanisms manipulating small amounts of data (e.g., 32 bit integers). To handle this specific case we ended up porting several cryptographic primitives to be run on the SCPU's main processor instead, and thus eliminate the hardware latencies for small data items. Space constraints prevent the discussion of the numerous other encountered challenges.

Overview. To remove SCPU-related storage limitations, the outsourced data is 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 engine are pulled in on demand by the Paging Module.

Query execution entails a set of stages. (0) In the first stage a client defines a database schema (and partially populates it). Sensitive attributes are marked, i.e., by deploying the "SENSITIVE" keyword that 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);
```

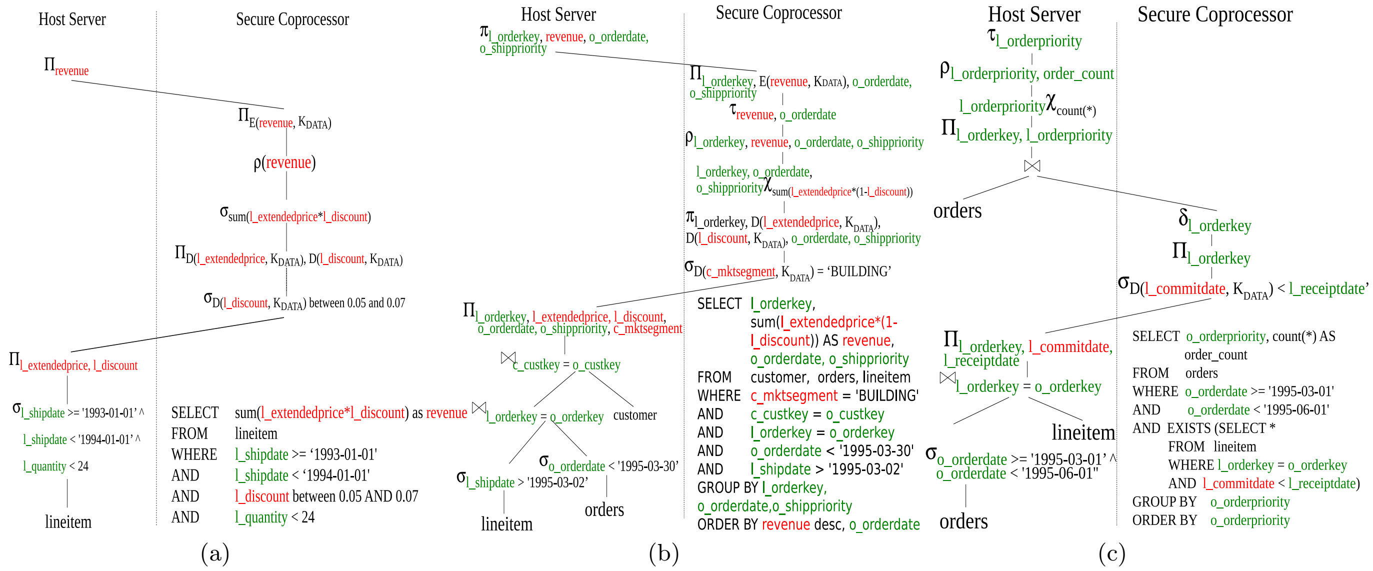
(1) Later, a client sends a query request to the host server through a standard SQL interface. The query is transparently encrypted at the client site using the public key of the SCPU. The host server thus cannot decrypt the query. (2) The host server forwards the encrypted query to the Request Handler inside the SCPU. (3) The Request Handler decrypts the query and forwards it to the Query Parser. The query is parsed and rewritten as a set of sub-queries, and, according to their target data set classification, each query is identified as being either public or private. (4) 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. (5) The final query result is assembled, encrypted (and digitally signed if correctness assurances are desired) by the SCPU database and the Query Dispatcher and sent back to the client.

3.2 Query Parsing

Outline. Sensitive attributes can occur anywhere within a query (e.g., in SELECT, WHERE or GROUP-BY clauses, in aggregation operators, or within sub-queries). The Query Parser's job is then:

- To ensure that any processing involving private attributes is done entirely within the SCPU. All private attributes are encrypted using shared data encryption keys between the client and the SCPU (Section 3.3), hence the host server cannot decipher these attributes.
- To optimize the rewrite of the client query such that most of the work is performed on the host server. This significantly increases performance.

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 among attributes based on security. Therefore, We define a specific attribute set classification into public


Host Server
Secure Coprocessor
Π revenue
Π E(revenue, K DATA)
ρ(revenue)
σ sum(l_extendedprice*l_discount)
Π D(l_extendedprice, K DATA), D(l_discount, K DATA)
σ D(l_discount, K DATA) between 0.05 and 0.07
Π l_extendedprice, l_discount
σ l_shipdate >= '1993-01-01' ^ l_shipdate < '1994-01-01' ^ l_quantity < 24
lineitem
Host Server
Secure Coprocessor
π l_orderkey, revenue, o_orderdate, o_shippriority
Π l_orderkey, E(revenue, KDATA), o_orderdate, o_shippriority
τ revenue, o_orderdate
ρ l_orderkey, revenue, o_orderdate, o_shippriority
l_orderkey, o_orderdate, o_shippriority χ sum(l_extendedprice*(1-l_discount))
π l_orderkey, D(l_extendedprice, K DATA), D(l_discount, K DATA), o_orderdate, o_shippriority
σ D(c_mktsegment, K DATA) = 'BUILDING'
Π l_orderkey, l_extendedprice, l_discount, o_orderdate, o_shippriority, c_mktsegment
⋈ c_custkey = o_custkey
⋈ l_orderkey = o_orderkey
customer
σ o_orderdate < '1995-03-30'
σ l_shipdate > '1995-03-02'
orders
lineitem
Host Server
Secure Coprocessor
τ l_orderpriority
ρ l_orderpriority, order_count
l_orderpriority χ count(*)
Π l_orderkey, l_orderpriority
⋈
orders
δ l_orderkey
Π l_orderkey
σ D(l_commitdate, K DATA) < l_receiptdate'
Π l_orderkey, l_commitdate, l_receiptdate
⋈ l_orderkey = o_orderkey
lineitem
σ o_orderdate >= '1995-03-01' ^ o_orderdate < '1995-06-01"
orders

```
SELECT    sum(l_extendedprice*l_discount) as revenue
FROM      lineitem
WHERE     l_shipdate >= '1993-01-01'
AND       l_shipdate < '1994-01-01'
AND       l_discount between 0.05 AND 0.07
AND       l_quantity < 24
```

(a)

```
SELECT  l_orderkey,
        sum(l_extendedprice*(1-
        l_discount)) AS revenue,
        o_orderdate, o_shippriority
FROM    customer,  orders, lineitem
WHERE   c_mktsegment = 'BUILDING'
AND     c_custkey = o_custkey
AND     l_orderkey = o_orderkey
AND     o_orderdate < '1995-03-30'
AND     l_shipdate > '1995-03-02'
GROUP BY l_orderkey,
o_orderdate,o_shippriority
ORDER BY revenue desc, o_orderdate
```

(b)

```
SELECT  o_orderpriority, count(*) AS
        order_count
FROM    orders
WHERE   o_orderdate >= '1995-03-01'
AND      o_orderdate < '1995-06-01'
AND  EXISTS (SELECT *
        FROM  lineitem
        WHERE l_orderkey = o_orderkey
        AND  l_commitdate < l_receiptdate)
GROUP BY    o_orderpriority
ORDER BY    o_orderpriority
```

(c)

Figure 3: TrustedDB query plans for TPC-H queries (a) Q6, (b) Q3, and (c) Q4. Showing private attributes (in red) and public attributes (in green).

(non-encrypted) and private (encrypted) types. 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) are illustrated in Figure 3.

Aggregation Example. 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. For e.g., query $Q6$ of the TPC-H benchmark is processed as shown in Figure 3 (a) The host server first executes a public query that filters all tuples which fall within the desired *ship date* and *quantity* range, both of these being public attributes. The result from this public query is then used by the SCPU to perform the aggregation operation on private attributes *extended price* and *discount*. While performing the aggregation the private attributes are decrypted inside the SCPU. Since the aggregation operation results in a new attribute composing of private attributes it is re-encrypted before sending to the client. This encryption is also done (transparent to the client) within the SCPU.

Note that the execution of private queries depends on the results from the execution of public queries and vice-a-versa even though they execute in separate database engines. This sharing of intermediate results is made possible by the TrustedDB Query Dispatcher in conjunction with the Paging Module (figure 2).

Grouping Example. If the client query specifies a GROUP or ORDER BY on public attributes but the selection includes an aggregation of the private attributes, the grouping or sort operation is performed inside the SCPU. Figure 3 (b) illustrates this for the TPC-H query $Q3$. If the aggregation did not involve any private attributes then the host server performs all the GROUP BY and sorting operations.

Nested Queries. The case of nested queries is similar, yet additional care should be taken when computing execution plans to limit the amount of data transfer between the host server and the SCPU which may result in suboptimal performance. One such example is query $Q4$ of the TPC-H benchmark which includes a sub-query on a private attribute. The query plan illustrated in Figure 3 (c) runs the removal of duplicates on attribute *order key* within the SCPU. An alternative would be to perform this operation on the host server. The choice to do this in the SCPU is made to reduce the traffic over the PCI interface.

3.3 Security

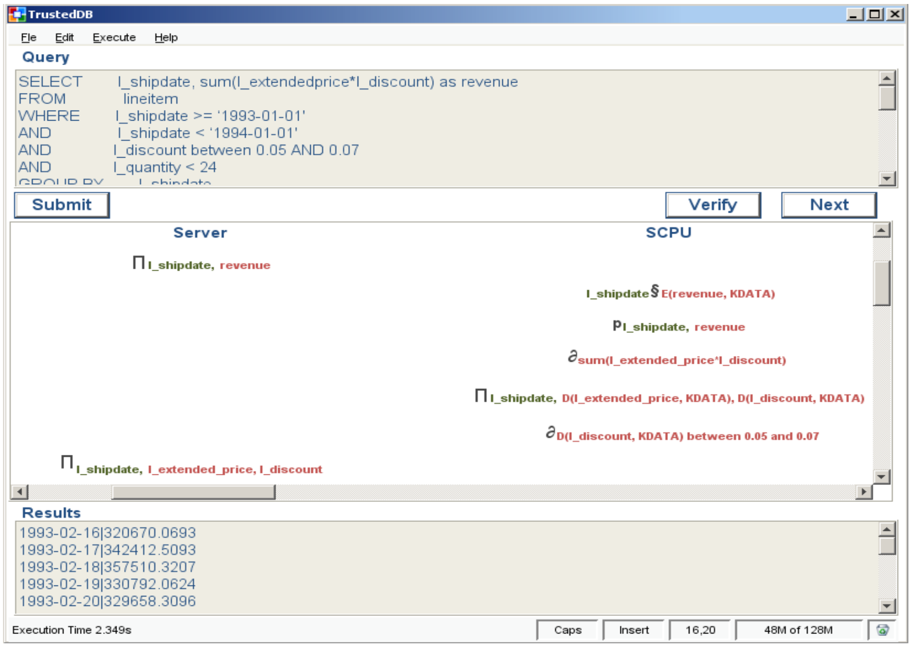
To cover all avenues of security clients need to be confident that (i) the remote SCPU was not tampered with and (ii) runs the correct TrustedDB code stack (including the correct user-land TrustedDB modules as well as the underlying OS and SCPU hardware logic). Finally, clients need to have the means to (iii) communicate secretly with the TrustedDB modules running inside the SCPU.

(i) is assured by the tamper-resistant construction of the SCPU which meets the FIPS 140-2 level 4 [1] physical security requirements. In the event of SCPU tamper detection, sensitive memory areas containing critical secrets are automatically erased. (ii) is ensured by deploying the SCPU *Outbound Authentication* [10] mechanisms. (iii) is achieved by deploying public-private key cryptography in key messaging stages. Both, client and the SCPU possess a public-private key pair (Figure 2). Messages sent between the client and the SCPU are encrypted [2].

Data Encryption.
For increased efficiency, fine-grained encryption of data is employed wherein, each individual attribute value within each tuple is encrypted separately with random keys generated by a cryptographic hash function based cipher initialized with K_{DATA} and per-tuple additional data that guarantees its uniqueness across the entire database. The result is based on a NMAC construction [5, 8]:

$$\begin{aligned} E(tbl.attr.val) &= ctr_{attr} \,||\, tbl.pri\_key \,||\, idx_K \,||\, (tbl.attr.val \oplus k) \\ k &= F(K_{DATA}[idx_K] \;\, || \, ctr_{attr} \,||\, tbl.pri\_key || F(K_{DATA}[idx_K])) \end{aligned} \tag{1}$$

[2] And thus, despite acting as a communication conduit between the client and the SCPU, the server cannot perform man-in-the-middle attacks and gain access to sensitive data.


TrustedDB
File Edit Execute Help
Query
Submit
Verify
Next
Server
SCPU
Π l_shipdate, revenue
l_shipdate § E(revenue, KDATA)
ρ l_shipdate, revenue
∂ sum(l_extended_price*l_discount)
Π l_shipdate, D(l_extended_price, KDATA), D(l_discount, KDATA)
∂ D(l_discount, KDATA) between 0.05 and 0.07
Π l_shipdate, l_extended_price, l_discount
Results
Caps
Insert
16,20
48M of 128M

```
SELECT     l_shipdate, sum(l_extendedprice*l_discount) as revenue
FROM        lineitem
WHERE      l_shipdate >= '1993-01-01'
AND         l_shipdate < '1994-01-01'
AND        l_discount between 0.05 AND 0.07
AND        l_quantity < 24
GROUP BY      l_shipdate
```

```
1993-02-16|320670.0693
1993-02-17|342412.5093
1993-02-18|357510.3207
1993-02-19|330792.0624
1993-02-20|329658.3096
```

Execution Time 2.349s

Figure 4: TrustedDB Client.

where *tbl* is the table name, *tbl.attr* is the attribute to be considered, *tbl.attr.val* is the plaintext value of the current tuple, *tbl.pri_key* is the primary key of the current tuple in table *tbl*, ctr_{attr} is a unique identifying number associated with $tbl.attr$[3], idx_K is an index in a table of K_{DATA} keys which allows multiple such keys to exist simultaneously (and be refreshed periodically) for increased security, and $F(\cdot)$ is a cryptographic hash function (SHA,MD5) [5].

4. DEMONSTRATION

This demonstration will show how TrustedDB enables generalized query processing over encrypted data. The demonstration will cover

- Running queries, perform data manipulation and data querying over outsourced encrypted data.
- Visualizing the workload schedule between the host server and the secure coprocessor - a key to making the use of trusted hardware in query processing practical.
- Gauging the security mechanisms employed to ensure the execution of queries over sensitive data in a remote secure environment.

Using more complex examples from standard benchmarks such as the TPC-H we will also demonstrate how TrustedDB achieves query processing over encrypted data without limiting query expressiveness.

5. REFERENCES

[1] FIPS PUB 140-2, Security Requirements for Cryptographic Modules. Online at http://csrc.nist.gov/groups/STM/cmvp/standards.html#02,2001.

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

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

[4] IBM 4765 PCIe Cryptographic Coprocessor. Online at http://www-03.ibm.com/security/cryptocards/pciecc/overview.shtml, 2010.

[5] Mihir Bellare. New proofs for nmac and hmac: Security without collision-resistance. In *Advances in Cryptology, Lecture Notes in Computer Science*, volume 4117, pages 602–619. Springer-Verlag, 2006.

[6] Marten Van Dijk, Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. Fully homomorphic encryption over the integers. In Henri Gilbert, editor, *EUROCRYPT*, volume 6110 of *Lecture Notes in Computer Science*, pages 24–43. Springer, 2010.

[7] Rosario Gennaro, Craig Gentry, and Bryan Parno. Non-interactive verifiable computing: Outsourcing computation to untrusted workers. In Tal Rabin, editor, *CRYPTO*, volume 6223 of *Lecture Notes in Computer Science*, pages 465–482. Springer, 2010.

[8] O. Goldreich. *Foundations of Cryptography I.* Cambridge University Press, 2001.

[9] M. O. Rabin. Digitalized signatures and public-key functions as intractable as factorization. Technical Report TR-212, Cambridge, MA, USA, 1979.

[10] Sean W Smith. Outbound authentication for programmable secure coprocessors. Darmouth College, Technical Report TR2001-401. Online at http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.58.4066, 2001.

[11] Sumeet Bajaj and Radu Sion. TrustedDB: A Trusted Hardware based Database with Privacy and Data Confidentiality. In *Proceedings of the ACM SIGMOD Conference*, pages 205–216, 2011.

[3] For storage efficiency we don't want to use the entire *tbl.attr* value in the result.