プロジェクト名：TEEを用いたセキュアかつ高性能なデータベースシステムの開発
申請者名：福山将英

【提案プロジェクト詳細】

1 何を作るか

1.1 概要

クラウドの利用は増加の一途を辿っている。令和4年版情報通信白書 [1]によると、2020年の時点でクラウドサービス市場規模は3281億ドルと、2017年の1640億ドルと比較して2倍ほど拡大しており、今後も市場は拡大していくと考えられている。GoogleやMicrosoft、Amazonなどの大企業がクラウド事業を提供しており、それらは外部からの攻撃に対する様々な策を講じているため、安全であると考えられている。しかし、安心は満たされていない。なぜなら、クラウド事業者は自身が提供しているマシンの管理者権限を有しているため、彼らがデータの読み取りや改ざんを行うかもしれない、という疑念が残るからだ。クラウドを利用するときは当然、組織の重要なデータベースをクラウドにおいて運用するが、クラウド事業者等の管理者権限を有する者が不正アクセスを行っていないことを保証するのは難しい。

本提案では、メニーコア環境下で性能を発揮できる、セキュアかつ高性能なトランザクション処理システムを有するデータベースシステムを開発し、この問題を解決する。具体的には、Intel SGXによって提供される隔離実行環境であるEnclave内部に、トランザクション処理プロトコルのSilo、インデックス及び機密データを配置することにより、OSやハイパーバイザ、クラウド事業者からの読み取り、改ざんを防ぐ。

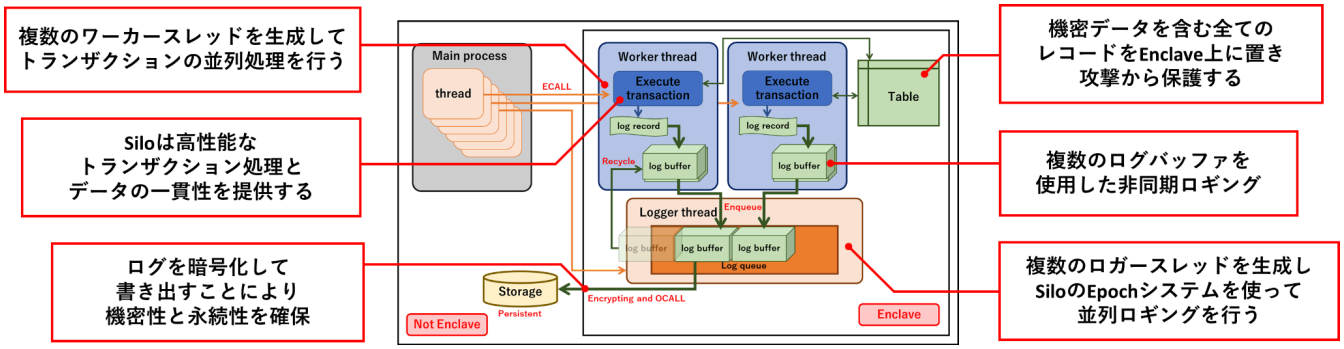


図1　システム概要図

本手法では、楽観的並行性制御法と並列ロギングを利用することによりトランザクション処理の性能を改善する。また、書き出したログの整合性を保証するためにログ改ざん検知プロトコルを実装する。これらによってデータの機密性、完全性を保証する。

近年の実証実験では、LayerXが提供するIntel SGXを基に開発されたAnonifyを用いて、インターネット投票における秘匿性と完全性を保証するというものがある [2]。このような技術は金融や連合学習など、処理性能を求められ、かつ情報資産を扱う分野において必要である。このプロジェクトは、データの機密性を保持しつつ処理性能を向上させることで、性能要求とセキュリティの確保のニーズを同時に満たしているため、情報資産を扱うあらゆる分野において応用が可能であると考えている。

1.2 背景

セキュリティの重要性

近年、クラウドコンピューティングの普及が加速し、今や組織の機密情報などが含まれているデータベースでさえも、クラウドに置いて運用するのが主流となっている。しかし、それはクラウド事業者を無条件に信頼するという前提が置かれている。従来のオンプレミス型であれば、外部からの不正アクセスに対する防御措置を取る必要があるが、クラウドではそれに加えてクラウド事業者自身が不正アクセスを行わないかを考慮する必

要がある。なぜなら、クラウド事業者は自身のマシンの管理者権限を有しているため、メモリ上のデータは理論上読み取ることが可能であるからだ。当然、機密情報は機密性を保証する必要があるが、クラウドコンピューティングでは上記の理由から機密性は保証されていない。

Confidential Computing

これを解決する技術として、準同型暗号を用いる秘密計算や、複数のサーバを用いる秘密分散が研究開発されてきた。これに加えて、近年立ち上がりつつある技術に機密計算(Confidential Computing)がある。機密計算とはデータ処理を暗号化した状態で行い、データの機密性を確保するというものである。機密計算はTrusted Execution Environment(TEE)と呼ばれる特殊なデバイスを用いる。TEEは現実的なパフォーマンスで機密性を保証できる技術として注目を集めている。TEEはハードウェア支援によって隔離実行環境を提供する。TEEの1つであるIntel SGXは、データを暗号化し、隔離実行環境であるEnclaveで保持する。データ処理を行う際は、CPU内にあるMemory Encryption Engine(MEE)によって復号化/暗号化され、CPUキャッシュ上のみ復号化される。上記の性質から、Intel SGXはCPUとCPU事業者であるIntelだけを信頼すればOSやハイパーバイザを信頼しなくてよい設計になっている。

従来研究・手法

データベースのセキュリティを確保する手法として様々な技術が提案されてきた。提案手法は大きく分けて2つあり、データを暗号化してそのまま処理する手法と、TEEを用いてデータの機密性を保証する手法である。

データを暗号化してそのまま処理する手法としては、CryptDB [3]やMONOMI [4]などが挙げられる。これらは準同型暗号や検索可能暗号、順序保存暗号をデータに適用し、暗号化したままデータの処理を行うというものである。しかし、暗号演算による法外なオーバーヘッドが発生し、実行できる命令にも制限がかかるため、汎用的なデータベースに適用するには難しい。

TEEを用いてデータの機密性を確保する手法としては、Always Encrypted [5]やShieldStore [6]、EnclaveDB [7]などが挙げられる。これらはTEEであるIntel SGXの機能を活用してデータの機密性を保証するというものである。しかし、旧来型の並行性制御法や逐次ロギングを採用していることから、スケールアップを行うことが難しい。

これらは総じて、今後の分散環境下におけるスケールアップが難しい設計になっており、加えてOSSとして提供されていないため、アルゴリズムの改善、エッジデバイスへの応用などが不可能であることはもちろんのこと、クラウド事業者が機密性を保ったデータベースを提供することがそもそもできないのが現状である。

1.3 作るもの

以上のことを踏まえ、メニーコア環境下に置いてもスケーラブルであり、セキュリティが確保できる本提案は価値がある。本プロジェクトでは、メニーコア環境下で性能を発揮できるセキュアかつ高性能なデータベースシステムを開発する。

1.3.1 設計

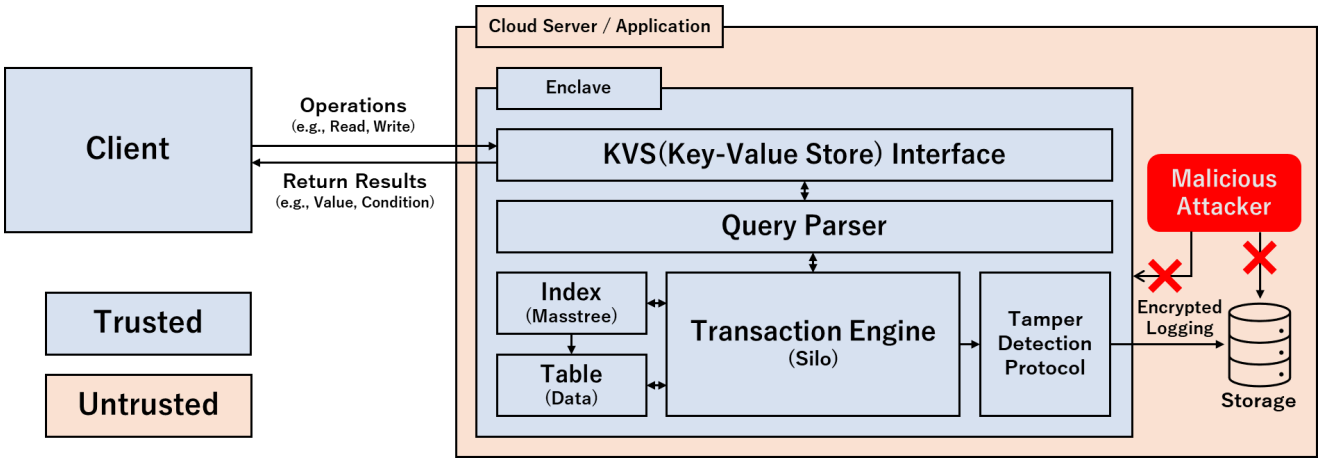


図2　本システムの概観

図2に本システムの概観を示す。本システムはKey-Value Storeとして開発を行う。データ構造は(*Key*, *Value*)のタプルであり、*Key*はString型、*Value*はint型で構成される。トランザクション処理エンジンとしてSilo [8]、データ検索用のインデックスとしてMasstree [9]を採用する。また、耐故障性を付与するためにロギング処理を行い、その完全性を保証するためにログ改ざん検知プロトコルを実装する。

Key-Value Store Interface

KVS Inferfaceはユーザーがデータに対して検索及び更新を行うために使用するものである。ユーザーはCLIにReadやWrite等の命令とその引数を入力し送信すると、Enclave内部に渡され、パーサーで解析、Siloエンジンでインデックスとテーブルに対して操作を行い、その結果をCLIに返却する。例えば、ユーザーが(“Hoge”, 100)を追加したい場合は、「Insert Hoge 100」と入力し、正常に処理が完了した場合は完了通知、既にKeyが使用されている場合はエラーを返しその旨を通知する。この一連の処理はEnclave内部で実行されるため、データの機密性が保証されている。使用できる命令としてRead, Write, Insert, Remove, Scanを実装する。これらの命令は必要に応じて1つ以上の引数を受け取り処理を行う。詳細は以下のとおりである。

- Read(*Key*)
  - Read命令は既にテーブルに存在する*Key*を検索し、その値を返却する。
  - 成功時の返り値：*Value*
  - 失敗時の返り値：ERROR(*Key*が存在しない、不適切な引数等)
- Write(*Key*, *Value*)
  - Write命令は既にテーブルに存在する*Key*に対応する*Value*を書き換える。
  - 成功時の返り値：SUCCESS
  - 失敗時の返り値：ERROR(*Key*が存在しない、不適切な引数等)
- Insert(*Key*, *Value*)
  - Insert命令はテーブルに新しく(*Key*, *Value*)のデータを追加する。
  - 成功時の返り値：SUCCESS
  - 失敗時の返り値：ERROR(*Key*が既に存在する、不適切な引数等)
- Remove(*Key*, *Value*)
  - Remove命令は既にテーブルに存在する(*Key*, *Value*)のデータを削除する
  - 成功時の返り値：SUCCESS
  - 失敗時の返り値：ERROR(*Key*が存在しない、不適切な引数等)
- Scan(*Key*, *number*)
  - Scan命令は*Key*を始めに*number*個分の(*Key*, *Value*)データを検索し、それらの*Key*と*Value*を返却する。

- 成功時の返り値：*number*個分の(*Key*, *Value*)
- 失敗時の返り値：ERROR(*Key*が存在しない、不適切な引数等)

Silo

Siloはメニーコアと大容量メモリを活用できるように設計されたトランザクション処理技法である。Siloは読み取りロックを取得せず、代わりに検証を行い整合性を保証する楽観的並行性制御であり、書き込みロックに関してもロックマネージャで集中管理せず、各データがロック状態を保持する非中央型ロック機構を採用している。これにより、全体のロック時間が減少するため、メニーコア環境でも高いスループットを達成することが知られている。また、Siloは並列ロギングが可能な設計になっているため、並列性を有する近代的ストレージデバイスを活用して性能低下を抑えることができる。

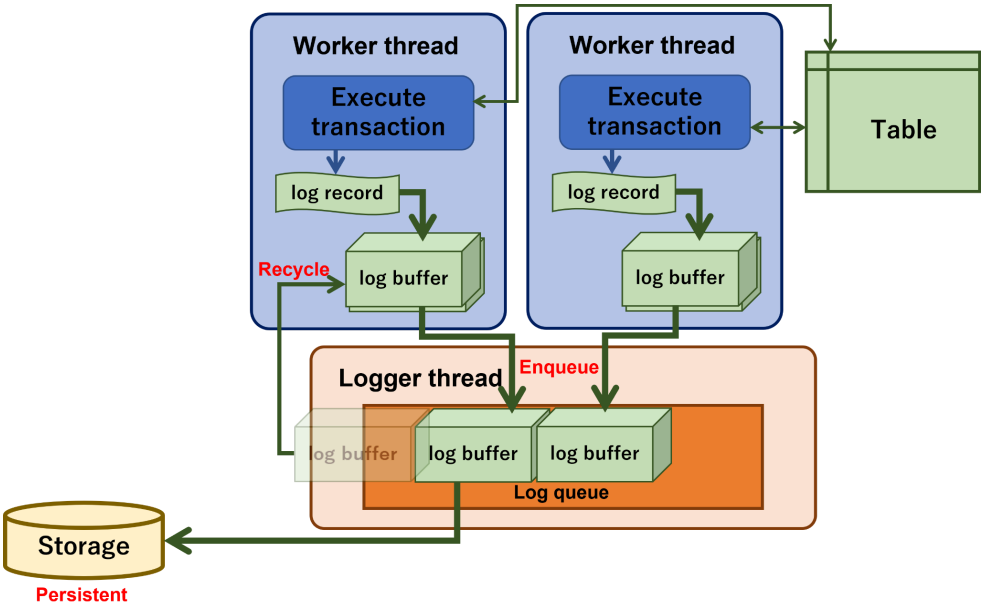


図3　Siloの概観

図3にSiloの概観を示す。Siloはトランザクションを処理するワーカースレッドと、ロギング処理を行うロガースレッドによって構成されている。各ワーカースレッドはテーブルのデータを用いてトランザクションを実行する。実行が完了しコミットフェーズに入ると、ワーカースレッドはトランザクションの書き込みリストに関するログレコードを生成し、ワーカースレッドが保有するログバッファに登録する。

ログバッファが一杯になるかエポックが更新されると、ワーカースレッドはログバッファを対応するロガースレッドのログキューに追加し、待機しているロガースレッドを起床させる。ワーカースレッドはログバッファを複数保有しており、ワーカースレッドはログバッファをロガースレッドのログキューに追加後、現在のログバッファを置き換えて処理を続行する。

起床したロガースレッドは、ログキューの先頭にあるログバッファをストレージデバイスに書き出し永続化を行う。永続化完了後、再利用できるようにログバッファを空にしてワーカースレッドに返却し、次の起床命令まで待機する。

ログ改ざん検知プロトコル

データベースシステムへの耐故障性を付与するためにトランザクションの結果をログとして出力しているが、復元時に全てのログが揃わないと一貫性のない復元が行われてしまい完全性が失われる。これを保証する手法として、ログの改ざん検知プロトコルを実装する。EnclaveDBではログを利用したハッシュ木を生成して整合性を検証しているが、並列ロギングではログ書き出し時に全てのログレコードが揃う設計ではないため、そのまま適用することができない。その為、並列ロギングに対応したログ改ざん検知プロトコルを実装する。

本システムを実現するために必要な実装物一覧

- KVSインターフェース
- Enclave内部で動作するMasstreeインデックス
- Enclave内部で動作するSiloプロトコル
- ログ改ざん検知プロトコル

1.3.2 到達目標

未踏プロジェクトでの本システムの到達目標は以下の通りである。

- SGXの仕様に適用したSiloプロトコルとMasstreeの実装を行う。
- KVSインタフェースでユーザーとの対話を可能にする。
- ログ改ざん検知プロトコルを実装し完全性の確保を行う。
- これらを隔離実行環境下で動かし機密性を保証する。

2 どんな出し方を考えているか

実装したシステムはOSSとして公開する。簡単な環境構築を行えば誰でもこのシステムの機能を使うことができる。SGXは隔離実行環境をシミュレートする機能を有しているため、必要な環境条件を満たしていなくても疑似動作を体験できる。

加えて、実装したシステムをMicrosoft、Googleに売り込むことを検討している。なぜなら、MicrosoftはConfidential ComputingとしてEnclaveDBを提案、GoogleはクラウドサービスとしてGoogle Cloud Spannerを提供しているが、本システムはEnclaveDBよりも高い分離レベルと並列ロギングによるスケーラビリティを提供し、Google Cloud Spannerで採用されているTwo-phase lockingよりも高速な並行性制御法のSiloを使用しているため、これらの既存手法よりも優位性があるからだ。

また、クラウドコンピューティング系の学会に投稿し、システムの認知及びブラッシュアップも検討している。

3 斬新さの主張、期待される効果など

3.1 斬新さの主張

前述の通り、本システムはMicrosoftのEnclaveDBやGoogleのGoogle Cloud Spannerよりも高性能、高分離レベル、スケーラビリティを有し、かつTEEを活用してデータの機密性を保証する。性能、データー貫性、セキュリティは全てトレードオフであると考えられてきたが、本システムはそれら全てを提供する。

3.2 期待される効果

クラウドサービス(Microsoft Azure、AWS、Google Cloud)はクラウド事業者が管理者権限を有しているため信頼できないが、TEEの機能を提供している場合、信頼できないクラウドでもTEE内部のデータベースは信頼できる。ただ、これらのサービスはOSSではないため、日本で同様のクラウド事業を立ち上げるにはコードがないため実現できない。そこで、本システムをOSSとして公開することにより、同様のサービスを実現することができるようになる。

また、本システムはOracleのMySQLやMicrosoftのEnclaveDB、GoogleのGoogle Cloud Spannerよりも高性能かつ安全性の高いサービスを提供できるため、これらのユーザーを根こそぎ獲得できるポテンシャルを秘めている。

4 具体的な進め方と予算

4.1 開発を行う場所

- 自宅

4.2 使用する計算機環境

- ノートPC(Intel(R) Core(TM) i7-1185G7 CPU / 16GB RAM / WSL Ubuntu 20.04)
- デスクトップPC(Intel(R) Core(TM) i5-10400F CPU / 64GB / WSL Ubuntu 20.04)

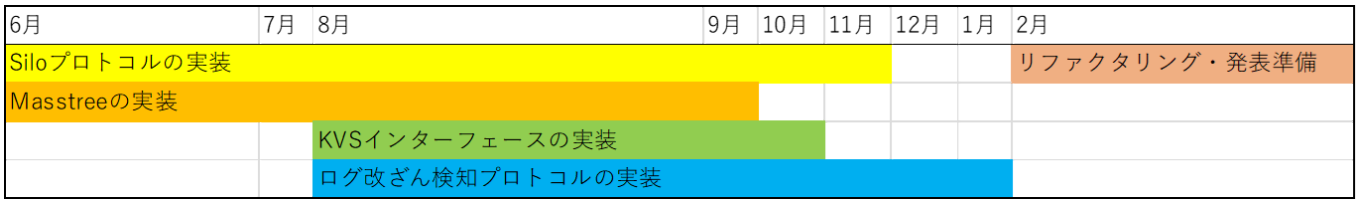
備考:SGXv1では隔離実行環境であるEnclaveで利用できるメモリ領域は128MBのみであり、Intel Core iシリーズとIntel Xeonシリーズの一部で提供されている。本システムはEnclaveのメモリ領域として512GBまで利用可能であるSGXv2を使用することを想定している。SGXv2はIntel Xeon 第3世代以降のプロセッサのみで提供されているため、開発は上記のもので行い、性能評価はSGXv2が利用可能なCPUを用いて行う予定である。

4.3 使用する言語、ツール

- C++
- SGX SDK(sgx_linux_x64_sdk_2.17.100.3 for Ubuntu 20.04-server)

備考:SGXの開発キットであるSDKはIntelが提供しているSGX SDKに加え、Rust-SGXやOpenEncalve等も存在する。SGX SDKを用いて開発を行う予定であるが、Rust-SGX、OpenEncalveに関しての調査が不十分であるため、開発期間中に調査を行い、SGX SDKよりも優位性、利便性が確認できた場合は開発キットを変更する可能性がある。

4.4 開発線表



4.5 開発に関わる時間帯と時間数

学部生であり、授業があるため正確な時間はわからないが、内部推薦で大学院進学予定であることと、卒業単位はほとんど取得済みであるため、若干フレックスになるが、ほぼ毎日フルタイムで開発時間を確保できる。

4.6 予算内訳をまとめた表

4.6.1 予算内訳

活動時間	活動時間×時給
1440時間	2,736,000円

4.6.2 必要経費

- 開発用PC(予算:500,000円)または、Intel Xeon 3rd gen CPUを搭載したPC(予算:1,200,000円)

5 提案者の腕前を証明できるもの

- eSilo
  - 開発言語:C++
  - Source: https://github.com/Noxy3301/enclaveSilo

Enclave内で動作するSiloプログラムである。SiloプログラムはCCBenchのものを参考にしているが、SGXはOSを信頼しない設計から、標準ライブラリやシステムコール等が利用できない仕様になっている。これに適応するように再実装、一部非対応関数(e.g., chrono、condition_variable)の再実装を行った。本プロジェクトはこのプログラムを基に開発を行う。
備考:第158回OS研究会に投稿した論文「SGXを用いたセキュアなSiloの設計」で性能評価に用いたプログラムであり、優秀若手発表賞を受賞した。

- B+tree
  - 開発言語:C++
  - Source: https://github.com/Noxy3301/bptree

データベースに用いられる多分木型インデックスの1つであるB+treeを実装したものである。

- IPX所内用半自動化ツール
  - 開発言語:Python

インターン先である弁理士法人IPXにて開発した自動化ツールである。OpenCV、pyautogui、selenium等を用いて、特許出願処理の一部自動化、拒絶査定時の引用文献自動ダウンロード等の機能を提供している。クリックする箇所を事前に画像として抽出しておき、現在のスクリーン上に一致する場所があるかをOpenCVを用いて探索し、該当箇所があった場合はpyautoguiを利用してクリック、ホットキー操作を行う。また、ブラウザ処理はseleniumを用いて要素検索を行い自動化を行っている。
備考:特許取得済みである。(特許7166700)

6 プロジェクト遂行にあたっての特記事項

川島研究会に在籍している。川島研究会主宰者の川島英之准教授は本事業への応募を了解している。

7 IT以外の勉強、特技、生活、趣味など

ソフトウェア作成以外の趣味としては、ゲームである。特に競技性の高いMOBAを好んで1日3～6時間はプレイする。リアルタイムで変わる戦況とそれに応じた作戦を考え、緻密に実行する能力が問われるため、一筋縄では上手くいかず、そこが非常に面白い点である。

勉強面では、高校生の頃から情報系に興味があり、趣味で基本情報技術者(2019年11月)と応用情報処理技術者(2021年12月)の資格を保有している。

8 将来のITについて思うこと・期すること

近年、Stable DiffusionやChatGPT、BingAIを台頭に、目まぐるしい技術革新が一般人にも可視化されるようになり、技術的特異点が着実に近づいてきていると言われている。このような技術は古来より積み重なる技術の結晶であり、現在も学会等で新規技術の手法に関する研究が活発に行われている。日本も例にもれず、このような学会や研究会が盛んに行われているため、これからも絶え間ない変化が起き続けることに胸を躍らせている。

しかし、日本のIT産業における競争力は低迷の一途を辿っているのにもかかわらず、IT産業の礎となる研究活動に十分な資金が配分されていない、適切な場所に配分されていない現実に絶望を感じている。日本の将来を担うであろうアカデミアの人間に支払われる給与は一般的な社会人の給与と大差なく、より良い待遇を受けられる外資や大企業に優秀な人材が流れるのは至極当然であり、アカデミアに残る魅力が失われつつある。予算配分に関しても、配分を行う人間が研究内容を理解できていないため、その研究の成果や出版先、学会発表等ではなく、出版した本数のみで評価が決定しているのが現状である。私も日本の競争力を向上させる人材になりたいと思うが、教授数名の話を聞く限り、面白みに欠けるのが所感である。研究活動に理解のある人間が指揮を取れるようになることを切に願う。

参考文献

[1] 令和4年版情報通信白書: https://www.soumu.go.jp/johotsusintokei/whitepaper/ja/r04/pdf/n3600000.pdf

[2] VOTE FORとLayerX、インターネット投票を見据え、高い秘匿性と非改ざん性を備えた市民意見収集システムをつくば市で実証へ: https://www.anonify.layerx.co.jp/post/pr20210921

[3] R. Popa, C. Redfield, N. Zeldovich, and H. Balakrishnan, “Cryptdb: protecting confidentiality with encrypted query processing,” in SOSP, 2011, pp. 85–100.

[4] S. Tu, M. F. Kaashoek, S. Madden, and N. Zeldovich, “Processing analytical queries over encrypted data,” PVLDB, vol. 6, no. 5, pp. 289–300, 2013.

[5] Panagiotis Antonopoulos, Arvind Arasu, Kunal D Singh, et al. 2020. Azure SQL Database Always Encrypted. In Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data. 1511–1525.

[6] Taehoon Kim, Joongun Park, Jaewook Woo, Seungheun Jeon, and Jaehyuk Huh. 2019. ShieldStore: Shielded In-memory Key-value Storage with SGX. In Proceedings of the European Conference on Computer Systems (EuroSys’19). 1–15.

[7] Christian Priebe, Kapil Vaswani, and Manuel Costa. 2018. EnclaveDB: A Secure Database Using SGX. In Proceedings of the IEEE Symposium on Security and Privacy (SP’18). IEEE, 264–278.

[8] Tu, S., Zheng, W., Kohler, E., Liskov, B. and Madden, S.: Speedy transactions in multicore in-memory databases, SOSP, pp. 18–32 (2013).

[9] Y. Mao, E. Kohler, and R. Morris. Cache craftiness for fast multicore key-value storage. In Eurosys, 2012.