

# Software-Defined な認証メモリ暗号化アーキテクチャの ボトルネック解析

金澤 颯飛<sup>1,a)</sup> 尾崎 巧基<sup>1</sup> 高前田 伸也<sup>1,2,b)</sup>

**概要:** コンピュータシステム上でのデータの安全性確保がより重要な課題となっている中で、高信頼実行環境 (Trusted Execution Environment, TEE) が広く利用されている。TEE はより重要なデータや処理を安全に取り扱うことができる技術であるが、実行環境によってはシステムに対する物理的な攻撃を考慮しなくてはならず、それゆえに DRAM メモリ上のデータ保護はより安全な TEE 技術の実現のための重要な課題の一つである。認証メモリ暗号 (Authenticated Memory Encryption, AME) 技術は、メモリ上のデータに対する物理的な読み取りと改竄の両方から、データを保護することができる暗号化技術である。従来の AME アーキテクチャに存在する性能オーバーヘッドと柔軟性の問題を同時に解決するためのアーキテクチャとして、Software-Defined な認証メモリ暗号 (SD-AME) アーキテクチャを提案したが、現状のアーキテクチャには高速化の余地が存在すると考えられる。本稿では、この提案した SD-AME アーキテクチャについて、シミュレーションによる性能解析を行い、また SD-AME アーキテクチャのパラメータのうちいくつかを変更してシミュレーションを行うことで、現状の SD-AME アーキテクチャの性能オーバーヘッドの要因を明らかにすることを目的とする。

## 1. はじめに

コンピュータシステムの取り扱う処理と情報の多様化が進み、コンピュータが個人情報をはじめとしたより重要で機密性の高いデータを扱うようになり、コンピュータシステム上でのデータの安全性確保は、より重要な課題となっている。これらのようなセキュリティ課題の現実的な解決策の一つとして、高信頼実行環境 (Trusted Execution Environment, TEE) が広く利用されている。

TEE は、OS と独立した安全な実行環境を提供することで、より重要なデータや処理を安全に取り扱うことができる技術である [1]。Secure な環境とそうでない環境を独立させて動作させる Arm TrustZone[2] や、enclave と呼ばれるプロセス内での実行を暗号化によって保護する Intel Software Guard Extensions(SGX)[3] など、その手法はプラットフォームによって異なるが、隔離された安全な実行環境を提供する。

一方で、実行環境によっては、コンピュータアーキテクチャに対する物理的な攻撃を考慮しなくてはならない。メ

モリなどのアーキテクチャ上において、プローブなどを用いて物理的にチップの持つデータを観測したり、改竄を行う攻撃が存在し、TEE 上においても、(Off-Chip な) メモリ上のデータはこれらの標的とされる [4], [5]。メモリ上のデータ保護は、より安全な TEE の実現のための重要な課題の一つである。

メモリ上のデータを保護するための技術の一つとして、認証メモリ暗号 (Authenticated Memory Encryption, AME) がある。認証メモリ暗号は、メモリの機密性と完全性を同時に保護する暗号化技術である。メモリ上のデータに対して暗号化を行い、その整合性を認証木によって検証することで、メモリ上のデータに対する物理的な読み取りと改竄の両方から、データを保護することができる。

認証メモリ暗号技術において、データを保護するための主な手法は、次の3つである：

**暗号化.** メモリ上に書き込まれるデータを暗号化することで、物理的な読み出しを行った際に情報が得られないようにする。

**MAC タグによる完全性確保.** ハッシュ関数を用いて、データから一意な MAC(Message Authentication Code) タグを生成し、データの完全性を確保する。

**認証木による整合性確保.** データを葉とする認証木を構築することで、MAC タグによる検証では検知できな

<sup>1</sup> 東京大学  
The University of Tokyo, Tokyo 113-8656, Japan

<sup>2</sup> 理化学研究所  
RIKEN

a) cs25\_ortlinde@is.s.u-tokyo.ac.jp

b) shinya@is.s.u-tokyo.ac.jp

い、リプレイ攻撃などの改竄を検知し、データの整合性を確保する。

商用 TEE における認証メモリ暗号技術は、Intel SGX において初めて導入されたが、現在の大容量な DRAM メインメモリを保護できるようなシステムは現状では存在しない [6]。Intel SGX の認証メモリ暗号は、保護可能領域が 128MB と小さく、また性能オーバーヘッドが厳しいものであったため、後にメモリ暗号化機構は認証木を持たないものになってしまった。

以上のような性能問題に加え、認証メモリ暗号技術には、柔軟性の問題も存在する。TEE アーキテクチャは、その実装や機能が CPU によって大きく異なり [1]、そのデータ構造や要求されるセキュリティ要件も異なってくる。ハードウェアによって実装される認証メモリ暗号アーキテクチャは、特定の要件に最適化されていることが多く、同一のアーキテクチャを異なる要件に適用することは難しい。一方で、柔軟性を求めてソフトウェアによる実装を行うと、性能オーバーヘッドが大きくなってしまう。

認証メモリ暗号の性能と柔軟性の問題を同時に解決するためのアーキテクチャとして、Software-Defined な認証メモリ暗号 (SD-AME) アーキテクチャ [7] を提案した。提案した SD-AME アーキテクチャの主な特徴は、次の通りである：

**暗号化機構のハードウェア実装。** 暗号化に用いる AES モジュールや XOR 演算モジュールなどの暗号化機構ハードウェアを搭載する。これにより、ソフトウェア実装に比べて高速な暗号化計算を実現することができる。

**ソフトウェア制御による柔軟性。** AME アーキテクチャ内に、AME アルゴリズムの実行を制御するためのコントローラを搭載する。これにより、AME アルゴリズムの変更や、CPU の変更に伴うデータ構造の変更などに、ハードウェアの変更を伴うことなく対応することができる。

**より高速な認証木。** 整合性検証木に Bonsai Merkle Tree[8] を採用し、高効率な整合性検証を行う。Bonsai Merkle Tree は、データ本体ではなく、MAC タグの Hash 計算においてデータと共に用いるカウンタの値を葉として用いる。これにより、データ全体を用いる通常の認証木に比べて、速度や認証木サイズ等の性能オーバーヘッドの小さい認証木を実現することができる。

SD-AME アーキテクチャは、認証メモリ暗号技術に存在する問題を解決するより優れた AME アーキテクチャを提供することが期待されるが、性能オーバーヘッドの問題が完全に解決されるわけではなく、また現状のアーキテクチャには高速化の余地が存在すると考えられる。本稿では、この提案した SD-AME アーキテクチャについて、シミュ

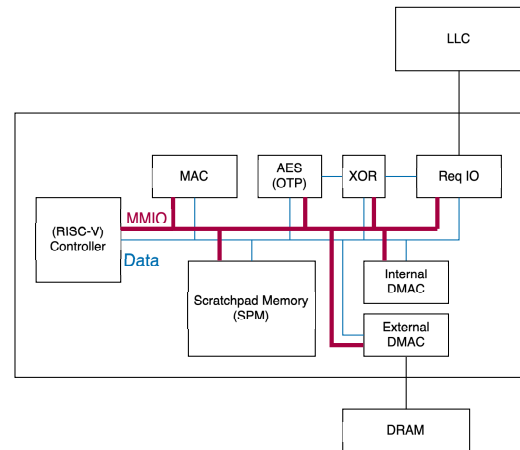


図 1 Software-Defined な認証メモリ暗号アーキテクチャの概略図

図 2 Software-Defined な認証メモリ暗号アーキテクチャの概略図

レーションによる性能解析を行い、SD-AME アーキテクチャのボトルネックの解析を行うことで、現状の SD-AME アーキテクチャの性能オーバーヘッドを明らかにし、今後の SD-AME アーキテクチャの高速化を目的とする。

## 2. Software-Defined な認証メモリ暗号アーキテクチャ

本章では、Software-Defined な認証メモリ暗号 (SD-AME) アーキテクチャ [7] の概要について述べる。2.1 節で SD-AME アーキテクチャの全体像を説明し、2.2 節でハードウェアの役割について、2.3 節でソフトウェアの役割について述べる。また、SD-AME アーキテクチャのプログラミングモデルについて述べる。

### 2.1 全体像

図 2 に、Software-Defined な認証メモリ暗号アーキテクチャの全体像を示す。本アーキテクチャは、RISC-V コントローラとソフトウェアのための命令メモリ、暗号化ハードウェア群、インターフェース回路、スクラッチパッドメモリ (SPM) 及び DMA コントローラから構成される。本アーキテクチャの最大の特徴は、ソフトウェア制御によって、認証メモリ暗号化アルゴリズムや、暗号化するデータ構造の変化に柔軟に対応できる点である。以下、それぞれの構成要素について、ソフトウェアとハードウェアに分けて説明する。

### 2.2 ハードウェア

#### 2.2.1 RISC-V コントローラ

AME ソフトウェアを実行するための環境として、RISC-V 64bit コントローラを採用する。コア上で行われる計算に浮動小数点演算を必要としないため、RV64I 命令セットを採用している。

### 2.2.2 暗号化演算ユニット

暗号化演算を行うためのハードウェアとして、AES モジュールと、XOR 演算モジュールを搭載する。AES モジュールは、暗号化に必要な One-Time Pad(OTP) の生成を行う。暗号化には、ソフトウェア制御によって渡されるカウンタの値と、ハードコードされている AES 秘密鍵を用いる。XOR 演算モジュールは、AES モジュールが生成した One-Time Pad と、ソフトウェア制御によって渡されるデータを XOR 演算して、暗号化されたデータを生成する。

### 2.2.3 スクラッチパッドメモリ (SPM)

AME アルゴリズムの実行に必要なデータや、メタデータを格納するための共有メモリ領域として、スクラッチパッドメモリを搭載する。MMIO アドレス空間にマッピングすることで、RISC-V コントローラ上のソフトウェアから load/store 命令を用いてアクセスすることができるようにする。

### 2.2.4 Request I/O

SD-AME アーキテクチャが Last Level Cache(LLC) からリクエストを受け取り、またレスポンスを送信するためのインターフェース回路を搭載する。このインターフェース回路は、LLC から各種データを含むリクエストを受け取り、MMIO レジスタにリクエスト情報を書き込む機能と、暗号化および復号化の完了を検知して、LLC へレスポンスを送信する機能を備える。また、必要な平文データの送受信を行うため、XOR 演算モジュールとの間にデータ転送路を有する。

### 2.2.5 DMA コントローラ

外部 DRAM 及び、内部 SPM に対する Direct Memory Access(DMA) を行うための DMA コントローラを搭載する。実際の DMA アクセスの管理に加え、DMA の完了通知のため、完了した DMA リクエストの ID を MMIO レジスタに書き込む機能も備える。

## 2.3 ソフトウェア

ソフトウェアの主な役割は、コントロールプレーンとデータプレーンの 2 つに分類される。

### 2.3.1 コントロールプレーン

認証メモリ暗号化計算の手順となるアルゴリズムをはじめとした、暗号化や認証およびそれに必要なデータ等の管理の手順をソフトウェアによって制御する。これには、次のようなものが含まれる：

**AME アルゴリズムのコントロール。** メタデータのためのアドレス計算や、AME アルゴリズムの実行を制御する。認証木のノード、カウンタ、データの MAC タグなどのメタデータのためのアドレス計算手法は、AME アルゴリズムの形態によって異なる。アドレス計算をソフトウェアによって制御することで、AME アルゴリズムの変更だけでなく、それに伴うデータ構造の変

更にも、ハードウェアを変更することなく対応することができる。

**メタデータの管理。** 暗号化や MAC タグ生成のためのメタデータの生成および、カウンタの制御をソフトウェアによって行う。

**コントローラ上のデータ管理。** On-Chip Buffer 上に保存するメタデータや、キャッシュデータの状態管理及び書き戻し手順などをソフトウェアによって定義する。

### 2.3.2 データプレーン

暗号化計算などの実行はハードウェアで行われるが、その入出力制御やデータ移動の管理は、ソフトウェアが担う。これには、主に次のようなものが含まれる：

**暗号化計算。** XOR, AES をはじめとした暗号化に必要な計算の実行制御を行う。

**データ転送。** SPM や DRAM, 各計算ユニット間のデータ転送の管理を、RISC-V コントローラ上のソフトウェアから行う。

### 2.3.3 プログラミングモデル

以上の役割を持つソフトウェアを、RISC-V コントローラ上で動作させる。ソフトウェアは、主に次のようなアーキテクチャ制御を行う：

**データキャッシュ。** SPM 上に、AME ソフトウェアで管理されるメタデータキャッシュを構築する。ソフトウェアから DMA コントローラへの MMIO を通じて、外部 DRAM と SPM の間で、メタデータの転送を行う。DMA リクエストは ID によって管理され、完了した DMA リクエストの ID を保持することで、DMA リクエストの管理を行う。

**SPM 管理。** 認証木のノードや、MAC タグなどのデータを SPM 上に保存するにあたっての管理を行う。本アーキテクチャにおいて、SPM は暗号化ユニットと DRAM の間のステージング領域として機能させる。これにより、SPM 上のデータの Write Back を、認証木のノードの更新などが完了したことをソフトウェアが確認してから行うことができ、ソフトウェアによるタイミング管理が可能になる。また、SPM 上のデータ状態 (Valid/Dirty など) の管理をメタデータキャッシュによって行う。

**ハードウェア回路の管理。** 暗号化モジュールへのメタデータの受け渡しなどを管理する。暗号化モジュールへのデータの受け渡しと、暗号化計算の開始の指示を、MMIO store 命令を用いて行う。

## 3. シミュレーションによる解析

本章では、前章で述べた SD-AME アーキテクチャをシミュレーションによって解析する手法について述べる。3.1 節で、SD-AME アーキテクチャのシミュレーションに用いるシミュレータやベンチマークについて説明し、3.2 節で、

具体的な解析手法について述べる。

### 3.1 セットアップ

本研究では、SD-AME アーキテクチャのシミュレーションを行うため、Spike Simulator および The Championship Simulator(ChampSim) を用いた。本節では、3.1.1 項にて RISC-V の命令セットシミュレータである Spike Simulator について、3.1.2 項にて Trace-Based なコンピュータアーキテクチャシミュレータである ChampSim について概説する。また、3.1.3 項にてシミュレーションにて使用するベンチマークについて述べる。

#### 3.1.1 Spike Simulator

RISC-V ISA Simulator である Spike[9] は、RISC-V International によって提供される、RISC-V プロセッサの命令セットアーキテクチャ (Instruction Set Architecture, ISA) をシミュレートするオープンソースプロセッサシミュレータである。基本構造は C++ で記述されており、RISC-V プロセッサコアと、レジスタ、メモリ管理ユニット (MMU)、パイプラインなどのプロセッサコアの構成要素が実装されている。Spike は、整数演算及び浮動小数点数演算を含む RISC-V 命令セットをサポートしている。Spike Simulator の利点はその幅広い拡張性であり、RISC-V の各種拡張命令セット、ユーザ定義の拡張命令に加え、アドレスのマッピングを定義することで、MMIO を用いることもできる。詳細な実装については [9] を参照されたい。

#### 3.1.2 ChampSim

The Championship Simulator(ChampSim)[10] は、Gober, N. らによって開発されたコンピュータアーキテクチャシミュレータである。シンプルな構造と、軽量の動作、そして高いモジュール性が主な特徴である。ChampSim は、分岐予測器などを持つ (Out-of-Order) プロセッサモデルと L1/L2 キャッシュ、LLC、DRAM、Translation Lookaside Buffer(TLB) 及び DRAM コントローラを含むメモリ階層モデルを備えるプロセッサシミュレータである。

ChampSim は、プロセッサの動作をシミュレートするために、入力として、ベンチマークプログラムそのものではなく、その動作内容が記録された Trace と呼ばれるファイルを用いる。Trace を用いることで、高い再現性を持つ、高速なシミュレーション環境を実現している。

#### 3.1.3 ベンチマーク

ベンチマークには、SPEC CPU 2017[11] のベンチマークから、gcc コンパイラ及び Perl インタープリタベンチマークを用いる。各ベンチマークは、各言語で記述されたプログラムを入力として受け取り、コンパイルされた実行ファイルまたは実行結果を出力する。ChampSim 上では、これらのベンチマークの Trace を用いることで、ベンチマークプログラムの動作をシミュレータ上に再現し、アーキテク

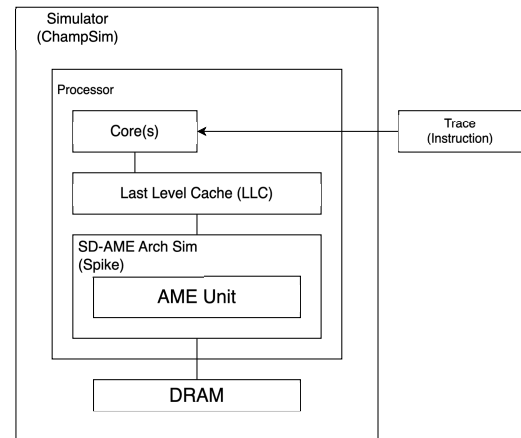


図 3 シミュレーションで用いるアーキテクチャの構成

チャの性能を評価する。本稿の評価において、各種 Trace は、The Third Data Prefetching Championship(DPC3) にて使用・配布されたものを用いた。

### 3.2 解析手法

SD-AME アーキテクチャを Spike 上に実装し、ChampSim 上でベンチマークプログラムを動作させることで、ベンチマークプログラムの動作に伴う DRAM アクセスと、SD-AME アーキテクチャによる認証メモリ暗号化をシミュレートする。図 3 に、本稿のシミュレーションで用いるアーキテクチャ全体の構成を、表 1 に、本稿で用いるアーキテクチャの主なパラメータを示す。基本構成では、2GHz シングルコアプロセッサに、32KB の L1 キャッシュ、1MB の L2 キャッシュ、2MB の LLC と 128KB のメタデータキャッシュを搭載したコンピュータアーキテクチャの、LLC と外部 DRAM の間に、SD-AME アーキテクチャに相当する RISC-V 64bit コントローラを搭載したアーキテクチャを接続することで、認証メモリ暗号ユニットを持つアーキテクチャを再現し、性能評価を行う。

#### 3.2.1 パラメータ

本稿では、SD-AME アーキテクチャのボトルネック解析を行うために、SD-AME アーキテクチャの性能に影響すると考えられるパラメータの一部を変更してシミュレーションを行う。以下に、本稿にて解析を行う SD-AME アーキテクチャのパラメータと、その目的を示す。

**動作速度。** SD-AME アーキテクチャ全体の動作速度を変更する。このアーキテクチャ全体には、RISC-V コントローラの周波数だけでなく、MMIO、DMA の転送速度や、暗号化ユニットの処理速度及びレイテンシを含む。シミュレータ上では、外部アーキテクチャ (ChampSim) の 1 サイクルの動作に対する SD-AME アーキテクチャ (Spike) の動作サイクル数を変更することで、SD-AME アーキテクチャ全体の外部アーキテクチャに対する相対的な動作速度を変更する。これ

表 1 主なシミュレーションパラメータ

外部 (ChampSim) シミュレータ	
コア	1 core, 2GHz, Out-of-Order
L1I Cache	32KB, 8-way, 4 cycle latency
L1D Cache	48KB, 12-way, 5 cycle latency
L2 Cache	512KB, 8-way, 10 cycle latency
LLC	2MB, 16-way, 20 cycle latency
Off-Chip メモリ (ChampSim)	
DRAM	DDR4-2400, 8GB
SD-AME アーキテクチャ (Spike)	
AES Latency	40 cycles
MAC Latency	40 cycles
Metadata Cache	128KB
Controller ISA	RISC-V 64bit

により、SD-AME アーキテクチャとその外部 (DRAM や LLC) とのデータ転送速度の影響を評価する。

**RISC-V コントローラのコア数.** RISC-V コントローラを複数準備し、SD-AME アーキテクチャをマルチコア化する。これにより、SD-AME アルゴリズムの並列化による高速化の影響を評価する。

本稿では、以上の2項目についてシミュレーションを行い、外部アーキテクチャで動作するベンチマークプログラムの動作速度から、SD-AME アーキテクチャの速度性能を評価する。

## 4. 解析結果

以下に、シミュレーションによって得られたデータを示す。

### 4.1 SD-AME アーキテクチャ全体の高速化

SD-AME アーキテクチャ全体の動作速度を変更した際の、外部アーキテクチャの動作速度を示す。現状の SD-AME アーキテクチャでは、DRAM とのデータ転送において必ず SPM を経由させる構成となっており、SPM の動作速度や RISC-V コアの動作速度などのパラメータには、高速化の余地が残されていると考えられる。また、十分に高速化した条件で得られた解析結果からデータ転送の影響を評価するためには、SD-AME アーキテクチャを持たない構

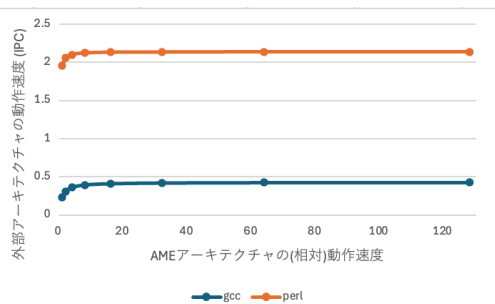


図 4 AME アーキテクチャ全体の動作速度を変更した際の外部アーキテクチャの動作速度

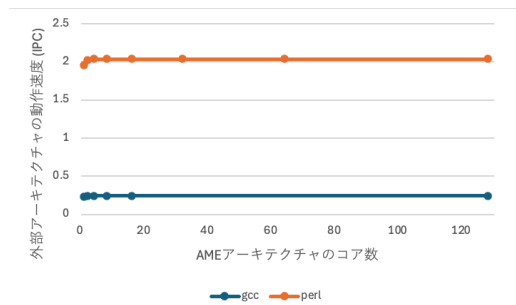


図 5 SD-AME アーキテクチャのコア数を変更した際の外部アーキテクチャの動作速度

成との比較や、データ転送路を変更した性能評価などが必要であり、当該パラメータ変更に対する性能評価には課題が残っている。

### 4.2 SD-AME アーキテクチャのマルチコア化

SD-AME アーキテクチャのコア数を変更した際の、外部アーキテクチャの動作速度を示す。

現状の SD-AME アーキテクチャの実装は、マルチコアによる並列化に十分に対応できておらず、並列動作時の DMA リクエスト ID の管理や、SPM 上のデータ状態の管理などに課題が残っている。今回の実験結果は、シングルコアとマルチコアの性能差から、マルチコア化による並列化の効果を示唆する一方で、より多くのコアを搭載する場合のアーキテクチャ設計には、なお解決すべき課題が残ることも示している。また、本解析では、一部パラメータでの実行時に、DMA リクエスト ID 管理に起因すると考えられるデッドロックの発生を確認した。これらの課題を解決するためのアーキテクチャ再設計と、DMA リクエスト ID 管理戦略の見直しが求められる。

## 5. まとめ

本稿では、Software-Defined な認証メモリ暗号 (SD-AME) アーキテクチャのシミュレーションによる性能解析を行った。本稿で解析したパラメータは SD-AME アーキテクチャの一部であり、ボトルネックとなり得る要素は、ここで示したもの以外にも存在する。より高性能な SD-AME アーキテクチャの実現に向け、引き続き本アーキテクチャの性能解析を進め、その結果に基づいて構成の改善を行う予定である。

## 謝辞

本研究の一部は、JST K Program JPMJKP24U4 の支援による。

## 参考文献

- [1] 須崎有康: Trusted Execution Environment の実装とそれを支える技術, 電子情報通信学会 基礎・境界ソサイエ

- ティ Fundamentals Review, Vol. 14, No. 2, pp. 107–117 (オンライン), DOI: 10.1587/essfr.14.2.107 (2020).
- [2] Pinto, S. and Santos, N.: Demystifying Arm TrustZone: A Comprehensive Survey, *ACM Comput. Surv.*, Vol. 51, No. 6 (online), DOI: 10.1145/3291047 (2019).
- [3] Costan, V. and Devadas, S.: Intel SGX Explained, Cryptology ePrint Archive, Paper 2016/086 (2016).
- [4] Chuang, J., Seto, A., Berrios, N., van Schaik, S., Garman, C. and Genkin, D.: TEE.fail: Breaking Trusted Execution Environments via DDR5 Memory Bus Interposition, *47th IEEE Symposium on Security and Privacy (IEEE S&P '26)*, IEEE Computer Society, (online), available from <https://tee.fail> (2026).
- [5] De Meulemeester, J., Oswald, D., Verbauwhede, I. and Van Bulck, J.: Battering RAM: Low-Cost Interposer Attacks on Confidential Computing via Dynamic Memory Aliasing, *47th IEEE Symposium on Security and Privacy (S&P)* (2026).
- [6] 中村草太, 松見湧斗, 内山一秀, 中條拓伯, 五島正裕: Intel SGX の認証メモリ暗号の性能評価, 第 256 回 ARC・第 211 回 SLDM・第 71 回 EMB 合同研究発表会 (2026).
- [7] 尾崎巧基, 金澤颯飛, 高前田伸也: Software-Defined な認証メモリ暗号アーキテクチャの提案, 第 255 回 ARC・第 202 回 HPC 合同研究発表会 (2025).
- [8] Rogers, B., Chhabra, S., Prvulovic, M. and Solihin, Y.: Using Address Independent Seed Encryption and Bonsai Merkle Trees to Make Secure Processors OS- and Performance-Friendly, *40th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO 2007)*, pp. 183–196 (online), DOI: 10.1109/MICRO.2007.16 (2007).
- [9] RISC-V International: riscv-software-src/riscv-isa-sim: Spike RISC-V ISA Simulator, RISC-V International (online), available from <https://github.com/riscv-software-src/riscv-isa-sim> (accessed 2026-04-22).
- [10] Gober, N., Chacon, G., Wang, L., Gratz, P. V., Jimenez, D. A., Teran, E., Pugsley, S. and Kim, J.: The Championship Simulator: Architectural Simulation for Education and Competition. (2022).
- [11] Standard Performance Evaluation Corporation: SPEC CPU 2017 Documentation, Standard Performance Evaluation Corporation (online), available from <https://www.spec.org/cpu2017/Docs/> (accessed 2026-04-22).