Side-channel Collision Attacks on Hyper-Dimensional Computing Based on Emerging Resistive Memories

Brojogopal Sapui
brojogopal.sapui@kit.edu
Karlsruhe Institute of Technology
Karlsruhe, Germany

Mehdi Tahoori
mehdi.tahoori@kit.edu
Karlsruhe Institute of Technology
Karlsruhe, Germany

ABSTRACT

Brain-inspired architectures are increasingly favored for edge devices due to their efficient execution of cognitive tasks with limited energy and computational resources. A promising approach in this field is Hyper-Dimensional Computing (HDC), known for its robustness against noise and simple computational operations, despite being constrained by memory bandwidth. HDC is well-suited for computation in memory (CiM) using emerging resistive memory technologies. However, security concerns arise from potential attack vectors in HDC, spanning from computational algorithms to the underlying technology. Since HDC relies on unique data patterns, or class hypervectors, stored in memory, there is a risk of undetected data manipulation or poisoning. We demonstrate that power information from insensitive (public) outputs can expose secret data stored in memories. This study investigates side-channel vulnerabilities in Content Addressable Memory (CAM)-HDC implemented with resistive memory-based CiM. We develop a collision attack using side-channel information to recover predicted classes from all possible outputs accurately. Our findings highlight a security threat in HDC even with parallel computation between query and class hypervectors. To address this vulnerability, we propose an effective countermeasure based on a hiding technique for CiM implementation, mitigating the identified security risks.

CCS CONCEPTS

• **Security and privacy** → **Side-channel analysis and countermeasures**.

KEYWORDS

side-channel, collision analysis, HDC, memristors, countermeasure

ACM Reference Format:
Brojogopal Sapui and Mehdi Tahoori. 2025. Side-channel Collision Attacks on Hyper-Dimensional Computing Based on Emerging Resistive Memories. In *30th Asia and South Pacific Design Automation Conference (ASPDAC '25), January 20–23, 2025, Tokyo, Japan.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3658617.3697768

1 INTRODUCTION

Biological computing systems prioritize efficiency over precision. Therefore, one strategy to reduce energy consumption in artificial systems involves adopting computational approaches that are inherently robust to uncertainty. Hyper-Dimensional Computing (HDC) is one of those frameworks that was initially introduced as a brain-inspired learning approach to achieve robust and efficient learning [19]. Over the past few years, HDC has found applications in various fields, such as language recognition [34], image classification [24], gesture recognition [33]. Compared to other machine learning algorithms, such as Deep Neural Networks (DNN), HDC demonstrates clear advantages, including smaller model sizes, reduced computational costs, one-shot learning, and resilience to noise and failure [29]. These qualities position it as a promising alternative for resource-constrained platforms.

HDC is based on the concept of hypervectors, i.e., vectors with a very large dimension [10]. In HDC, raw data (such as images, text, etc.) are encoded into high-dimensional vectors and rotate primarily around three fundamental operations, namely addition, multiplication, and permutation. With these three operations, HDC involves computing the similarity between two hypervectors i.e. query hypervector and class hypervector [35]. During the inference process, the system calculates the similarity of an unlabeled query hypervector to all class hypervectors, ultimately selecting the most similar class as the label. The hardware implementation of this inference operation is carried out by Associative Memory (AM) [1, 41]. However, the challenges posed by data transfer overhead and the hardware constraints due to the deceleration of transistor scaling can significantly obstruct the large-vector computation.

To address both challenges at the same time, the emerging Computation-in-Memory (CiM) paradigm proves to be a promising candidate [21]. The implementation of CiM extends to various memory technologies, ranging from traditional SRAM [18] to cutting-edge non-volatile resistive memories [42, 43]. Unlike conventional separation of data and computation, this paradigm involves enhancing memory to directly execute simple computations. Notably, in the literature, CiM-based AMs have been used in various applications [14, 28]. At the circuit level, cells based on content addressable memory (CAM) are used to realize AM [15, 21]. Resistive memories in CAM-based HDC offer low static power consumption and enhanced density [13]. Analog CiM can be employed to further enhance performance and energy efficiency [31], despite the trade-off of increased noise in computations.

The class hypervectors carry sensitive data in HDC. If these vectors are not secured, there is a risk of unauthorized access or tampering, which could compromise the integrity of the classification process [50]. Adversarial attacks on the class hypervectors [4, 46] can severely affect the accuracy and reliability of the HDC system in classifying new data. An adversary can also analyze side-channel information (such as power, electromagnetic radiations, etc. [36]) to

ASPDAC '25, January 20–23, 2025, Tokyo, Japan

ACM ISBN 979-8-4007-0635-6/25/01
https://doi.org/10.1145/3658617.3697768

extract the value of sensitive hypervectors from the HDC architecture. In traditional side-channel attacks[11] and collision power attacks [48] for cryptographic key recovery, attackers require a substantial number of power traces corresponding to the same input vector. This process often requires millions of traces, making data collection from physical devices costly and sometimes impractical. In CiM architectures, the challenge is further increased due to the integration of computation within memory arrays, necessitating a similarly large volume of traces for effective attacks [37]. These limitations highlight the need for more efficient methodologies or improved sensitivity in side-channel analysis to overcome the limitations of large-scale data collection.

In this paper, we present an extended side-channel attack model that requires only a few traces for high accuracy. Our method uses matching pattern collision analysis, starting with Hamming distance modeling of class hypervectors to identify target hypervectors. This reduces the samples needed for side-channel power analysis. We tested attacks on CAM-based HDC using ReRAM. By observing transient logic computation and using a divide-and-conquer algorithm, we accurately predict classes. Furthermore, the impact of process variation (PV), power delivery network (PDN), and measurement noise (MN) is observed to show an effect on the success of the attack. To counteract this, we propose a circuit-level hiding technique by duplicating the crossbar.

The remainder of the paper is structured as follows: section 2 presents the technological background of CAM-based HDC, CiM architectures, and side-channel collision attacks. In section 3, we describe our simulation and vulnerability analysis approach with an algorithm and power analysis. We explain the mitigated design in section 4. The results are presented and discussed in section 5. Finally, section 6 concludes this paper.

2 PRELIMINARIES

2.1 Principles of HDC, CAM, and CiM

2.1.1 HDC Cocnepts. HDC is a paradigm in which indices of object features and corresponding values are represented using hyperdimensional vectors (HV), where $HV \in \{1,-1\}^D$. An emerging application of HDC is in classification tasks. HDC-based classification involves three main steps: Encoding, Training, and Inference.

In the HDC encoding module, an input sample is represented as a feature vector $F = \{f_1, f_2, ..., f_N\}$ containing N features. The feature values are discretized into M levels, determined by the minimum and maximum values across the entire dataset. This discretization results in the construction of M correlated hypervectors [16]. The encoding module then represents the input feature F with a D-dimensional hypervector. The encoding scheme utilized is application-dependent and the literature offers several methods, together with data type analysis [10, 38].

With the input sample represented by the hypervector H, an HDC model can be trained as $ClassHV_j = _{H\in\omega_j} H$, where ω_j represents the hypervectors of the j-th object class assuming there are C classes in total. In binary HDC model, $ClassHV_j$ and H are both binarized data. During the inference stage, a query sample is first encoded into a hypervector $QueryHV$ using the same feature hypervectors. Subsequently, the similarity between $QueryHV$ and the hypervectors representing different classes ($ClassHV$s) is calculated as shown in Figure 1a. The class hypervector with the highest similarity to $QueryHV$ is then used to determine the inference label. In the binary model, the similarity between the $QueryHV$ and $ClassHV$s is quantified as the Hamming distance between them.

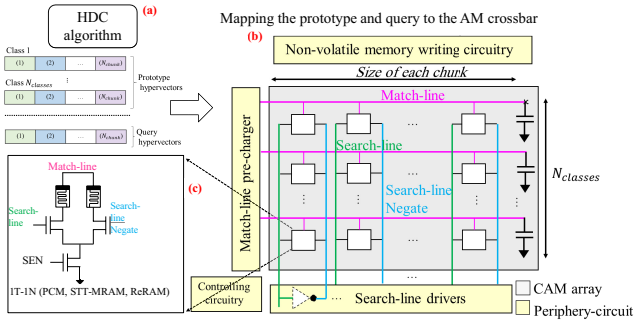


Figure 1: Overview of the HDC framework to be attacked. (a) Split the hypervectors into chunks at the algorithmic level. (b) The array-level realization of the CAM structure. (c) The differential structure of the CAM for 1T-1NVM (ReRAM).

2.1.2 HDC architecture with CAM. CAM is a type of memory architecture that enables rapid search and retrieval of data based on its content rather than its address [32]. CAM performs similarity matching by associating data with their content, allowing quick identification of matching patterns. CAM hardware as shown in Figure 1b), characterized by its high parallelism, finds applications in high-performance tasks such as HDC [13, 17, 23]. Each CAM operation involves two sets of data operands: $ClassHV$, which remains fixed, and $QueryHV$, which are simultaneously compared against all $ClassHV_j$. Content tags (or IDs) are associated with each $ClassHV$ entry and serve as unique identifiers based on the content of the stored data. During the comparison, if the content tag in a cell matches each bit of $QueryHV$, the corresponding match line is activated. Since $ClassHV$ does not change dynamically, it must only be written once in memory, resulting in almost no static power consumption [27].

2.1.3 CAM with Memristive Memory based CiM. Memristive memory based CAM cells are well suited for the specific use of HDC due to their high density and scalability [20–22]. Figure 1c illustrates the CAM cell based on memristive technology such as ReRAM (similar to PCM and STT-MRAM). In such CAM architecture originally proposed by [26], the original and complementary bits of the $ClassHV$ patterns are stored in the ReRAM-based memories, while the original and negated bits of the query pattern are applied as binary voltage levels to the search line and the negated search line, respectively. By interpreting binary '1' as a high-resistive state (HRS) and '0' as a low-resistive state (LRS) in the $ClassHV$ and using high and low voltage levels in the query patterns, a match is detected during the search operation. At the time of sampling, the voltage of the CAM match-line surpasses that of a mismatch. However, due to the limited HRS/LRS ratio in various memristive technologies, there are constraints on how much the length of the match line can be extended.

2.2 Collision Attack

2.2.1 Algorithmic Collision Analysis. Schramm et al. [39] proposed the internal collision analysis, which is based on the fact that an internal collision occurs when a function in the cryptographic algorithm delivers the same output corresponding to 2 different inputs $x1$ and $x2$. If the function and input are dependent on secret data, an attacker can acquire information about the secret data by collision analysis with the predefined model. During the computation of matching-based operations between queries and class hypervectors, if the Hamming distance information acquired between them is observed to be minimal (taking into account a certain amount of noise), it can be assumed that the corresponding two hypervectors have the maximum matching bits. If the Hamming distance information is zero, we can say that it matches exactly and that a perfect collision occurs. However, we do not always achieve an exact matching query. Therefore, we define the maximum matching conditions to extract sensitive information [9].

Suppose two vectors $w_1, w_2 \in 0, 1^D$ with $|w_1| = |w_2|$, the Hamming distance between w_1 and w_2 is defined as $d_{HAM} w_1, w_2 = |\{i \mid w_1 i \neq w_2 i \; \forall \; 1 \leq i \leq |w_1|\}|$. The Hamming distance, therefore, describes the number of mismatches between two vectors. For a pattern α and a vector w, we can similarly define the Hamming distance between α and w as $d_{HAM} \alpha, w = min\{d_{HAM} h\alpha, w \mid h$ is a substitution of the variables of $\alpha\}$. With these definitions, we can introduce two important matching problems which are relevant in this work. In the first problem, for a certain distance δ between the image data $h\alpha$ of the pattern α under substitution h and the target vector w instead of looking for an exact match. In the second problem, we are interested in finding a substitution h such that the number of mismatches between $h\alpha$ and the target word w is minimal for all possible choices of h. These are the most probable candidates to make a collision with image vectors and recover information about the image vectors.

2.2.2 Side-channel based Collision Analysis. In side-channel-based collision analysis, power traces are generated during the execution of the logic operation on a tamper-resistant device. Specifically, different random vectors are selected (denoted as w_1, ..., w_n) and for each vector, the data-dependent power consumption of the whole system is measured. The resulting power traces, namely (C_1, ..., C_n) with the help of a pre-assumed power model (such as Hamming distance model) can reveal sensitive information.

Given the substantial number of clock cycles in a logic operation in the power curve, we can reasonably assume that distinct input vectors will yield different power trace patterns. In the case of a matching pattern analysis, the unmatched input vectors give a higher Hamming distance value, but most matching vectors give the lowest Hamming distance value. Consequently, we can say that a collision occurs when the maximum match with the lowest Hamming distance between two vectors occurs. In this process, we can distinguish collisions and predict the target vectors.

2.3 Related Work

The existing studies on HDC classifiers have primarily focused on improving energy efficiency, inference latency, privacy preservation, or architecture design. Recent research [45] shows adversarial attacks on HDC classifiers implemented in a processor-centric architecture. The authors focus on a grey-box scenario in which the attacker is restricted from accessing and controlling sensitive parameters (such as position memory, value memory, and associative memory) as well as the training dataset of the targeted HDC classifier. However, the attacker is permitted to iteratively submit images to the HDC classifier and retrieve the associated prediction labels. The authors framed the attack as a regularized optimization challenge. The attacker's objective is to manipulate the target HDC classifier into generating incorrect prediction labels (executing a non-targeted attack) while minimizing the introduced perturbation.

Considering the significance of intellectual property (IP) in the HDC model, as discussed in a recent work [6], it is crucial to prioritize the protection of its fundamental hypervectors, such as feature and value hypervectors. The authors showed that failure to protect these base hypervectors could potentially empower an attacker to replicate a comparable HDC model for malicious activities, including but not limited to reverse engineering input or creating adversarial inputs. In that threat model, the authors presume that the raw data of the base hypervectors are susceptible, while the mapping information remains secure and inaccessible to attackers. Consequently, the attacker can solely access the unindexed hypervectors stored in the non-secured memory, allowing them to devise the inputs and observe the encoding outputs.

On the other hand, collision attacks have already been explored in conventional crypto key recovery. One of the earliest instances of a collision attack that can be applied to public key cryptographic algorithms is known as the doubling attack [8]. It necessitates that an adversary gathers traces of a suitable side channel during the execution of two scalar multiplications, with the input messages being deliberately chosen. When the side-channel information acquired during the computation of these doubling operations is observed to be identical (considering a certain amount of noise), it can be assumed that the two doubling operations have the same input point. This would allow an adversary to deduce the information about the bits which has zero value at every iteration. Yen et al. [49] expanded the above-mentioned attack to adapt it for use in a specific application[30]). The attack still relies on the same deliberately chosen inputs. However, side-channel-based collision attacks in CiM-based HDC have not yet been reported.

3 METHODOLOGY

Although HDC involves large-dimensional vectors and complex operations, it remains susceptible to adversarial attacks. The attack vectors depend heavily on factors such as hardware architecture and hypervector dimensions. Notably, reducing hypervector dimensions to less than 100 significantly degrades HDC inference accuracy [23].However, the scalability of the ReRAM is severely limited by the lower HRS-LRS ratio of ReRAM compared to SRAM, as well as the impact of manufacturing variability. In this section, we thoroughly discuss the hardware architecture and simulation methodologies essential for HDC, highlighting strategies for balancing scalability and robustness. We aim to evaluate the NVM-CAM structure for similarity computation due to its energy efficiency. We introduce a targeted attack method employing collision analysis, followed by a side-channel attack, to evaluate vulnerabilities of HDC accelerator.

3.1 Threat Model

Given the intellectual property (IP) value of the HDC model, it is crucial to ensure the robust protection of its foundational hypervectors such as class hypervectors. Failing to safeguard these base hypervectors could enable attackers to duplicate a comparable HDC model, facilitating malicious activities like reverse engineering of inputs or the generation of adversarial inputs. From an attacker's point of view, there are some limitations in the attack model that must be addressed before mounting a successful attack. In our threat model, the assumptions are as follows.

- The attacker is restricted from accessing and manipulating sensitive parameters such as index, value, and associative memory. The attacker is also not capable of accessing the training dataset of the target HDC classifier.
- For each input (image data), the attacker has access to both input data and can observe the output. The attacker retains the ability to iteratively submit input images to the HDC classifier and retrieve the associated prediction labels.
- The attacker is able to choose input vectors in a way to craft the required query vector, if the encoding is known.
- The attacker is allowed to introduce noise into the images.
- The attacker measures the total instantaneous power of the HDC accelerator (no access to power profile of submodules).

Given the restricted information about the target HDC classifier, the attacker's objective is to create an adversarial image with minimal perturbation. The aim is to mislead the HDC classifier by predicting a label different from the true label, aligning with the characteristics of general attacks in conventional adversarial machine learning [12]. In contrast to crypto key recovery, where precision is crucial bit by bit, HDC exhibits a distinctive characteristic: even if the class hypervectors are approximately recovered, it suffices. This is due to the fact that the training process is not exact in HDC [40]. Moreover, HDC is tolerant to small bit-flips because the information is evenly distributed over every bit of the hypervectors [10].

3.2 Simulation Approach

In literature, various architectural designs exist for CAM-based and CiM-based HDC inference engines mentioned in subsection 2.1. Most importantly, the design frameworks follow a similar approach to simulation, and our framework is in line with what many researchers have proposed. To validate our security claim, we simulate the entire circuit with the SPICE simulator as shown in Figure 1.

The generation of hypervectors that is generally implemented with bitwise XOR operation, known as encoding, occurs at the algorithmic level in our work. Selecting the data type and hypervector length is crucial and should be tailored to the specific hardware characteristics. We use binary data types because those are more compatible with the hardware. Furthermore, it is essential for the hypervectors to possess a substantial size to preserve high inference accuracy. We consider a maximum of 256-bit hypervectors in our CAM-based HDC.

To co-optimize the inference algorithm concerning the underlying hardware design and resistive memory technology, it is necessary to partition the large hypervectors into smaller chunks at the algorithmic level. The efficient mapping of these hypervectors chunks onto the memristive-CAM accelerator requires additional information, which the algorithm must furnish. The information serves to identify whether a chunk originates from a class or query and indicates its position within the original hypervector. The role of the HDC inference accelerator is to process these chunks in a manner that ensures that the final output of the similarity computation remains unaffected by the chunking process.

The key feature of the AM (Associative Memory) is the capability of parallel search execution. It is important to note that the number of match-lines in AM is equivalent to the number of classes. During each activation cycle of the AM architecture, the initial step involves performing analog similarity computations between the entire query and the class hypervectors. However, it is necessary to accumulate the Hamming distances across all the chunks to determine the most similar class to which the query should be assigned, i.e., the class with the lowest Hamming distance.

3.3 Overview of the Attack

The proposed attack is based on two steps: first, we attack the desired class hypervector by brute-forcing the input query hypervector byte-by-byte. This reduces the search space from large numbers to only a few candidates per byte, which we determine using a side-channel-based collision attack. In this section, we explain the two approaches in more detail. Note that we ignore the encoding part for the attack because of the assumption on chosen input vectors mentioned in 3.1.

As the classification output itself already provides information to the attacker to a certain extent, we first attack the class hypervector on an algorithmic level by merely observing the classification result. Classification in HDC is based on computing the Hamming distance to all class hypervectors. Therefore, the classification result already informs which class hypervector is closest to the current query hypervector. We leverage this to gather candidates for all class hypervectors of length n bytes as follows:

Algorithm 1 Hamming distance-based Collision Analysis

```
1:  procedure CollisionAttack(HDCModel)
2:      candidates ← initializeEmptyArray()
3:      for i ← 0 to n − 1 do
4:          for j ← 0 to 255 do
5:              setAllZeroqueryHV
6:              for k ← 0 to i do
7:                  queryHVk ← j
8:              end for
9:              if classifyHDCModel, queryHV = c then
10:                 candidatesci.append j
11:             end if
12:         end for
13:     end for
14: end procedure
```

The algorithm begins by initializing an empty array, `candidates`, designed to hold potential byte values for each class hypervector. The algorithm then performs an iterative examination of each byte position from 0 to $n-1$. For a given byte position i, the algorithm assigns all possible values (0 to 255) to that byte while setting all other bytes to zero, thereby generating a modified query hypervector. This modified query hypervector is then classified using the HDC model. If the classification result matches the target class c, the current byte

value j is recorded in the candidate list for the i-th byte of class c. This procedure is systematically applied to all byte positions and potential byte values, ultimately yielding a comprehensive set of candidate byte values for each class hypervector.

The described algorithm 1 results in a set of candidates for each class c, which will be further reduced to determine the correct values using power measurements. The remaining set of candidates is now small enough to be brute-forced again using the side-channel analysis. By measuring the power consumption of the CAM array during classification, we determine which of the possible query hypervectors for each class matches best. In the results, we provide the relation between power consumption and matching bytes compared to non-matching bytes. We observe the same when attacking the entire n-byte class hypervector using the remaining set of candidates. This allows an attacker to easily determine the correct class hypervector using power measurements.

4 PROTECTED CIM DESIGN

The objective of implementing a hiding countermeasure is to hide the data-dependent power consumption resulting from the logic operation during the search for matching bits. To hide the dependency of the power on the input query, we follow the dual-rail method to mitigate the threat. We observe that the power consumption is maximum if there is a bit-wise mismatch for all the bits in a query. Leveraging this dual characteristic, we propose a hiding countermeasure tailored to CiM.

In this work, we follow the concept of power equalization by adding duplicated logic that will mimic the inverse behavior of our actual circuit. We store each complemented resistive state of the cell of each class hypervectors to the extended dummy hypervectors. Thus, we double the class hypervectors among which half of them are actual hypervectors and another half of them are dummy class hypervectors. Before starting the HDC operation at the AM module in HDC, the dummy class hypervectors are stored with complemented state of the actual class hypervectors. During HDC operation, we activate the respective rows in both parts of the extended array and the query hypervectors perform a match operation with all hypervectors in parallel. This process complements each LRS activation with an HRS activation which equalizes the overall power. In our design, we extend the original 256×10 array with a complementary 256×20 array as shown in Figure 2.

In an ideal circuit, the resulting power consumption is equal for all input values, whereas our more realistic circuit contains a certain amount of process and design variations that also lead to runtime variations later on. These remaining variations result in side-channel leakage, which may eventually allow all bits to be extracted. However, the number of measurements required will increase significantly, which could be very hard to analyze.

5 RESULTS AND DISCUSSION

5.1 Experimental Setup

The CiM with ReRAM for the CAM-HDC crossbar (256×10) is implemented using a 22*nm* Global Foundry technology. We perform a Monte Carlo simulation for each state of the ReRAM with 1,000 samples fixed at room temperature to obtain the resistance distribution. The device model and further details are given in Table 1. We generate

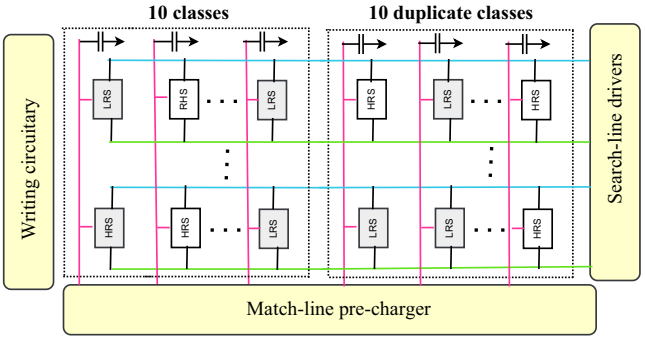


Figure 2: Proposed CiM array with protection against power analysis side-channel collision attack by column duplication

Table 1: Simulation setup tools and parameters.

Simulation tool	Cadence Virtuoso
Technology node	Global Foundries 22FDX
Standard V_{DD}	0.8 V
Temperature	27 °C
Interconnect parasitic	• 22 nm interconnect • $RC_{\pi-model}$ = 11.95 Ω, 16.63 aF (per CAM cell)
ReRAM model [44]:	• Filament radius = 45 nm • Disc region length = 0.6 nm • HRS, LRS = 105.51 kΩ, 1.975 kΩ

power traces from the HDC by performing circuit-level simulations in the Cadence Virtuoso. We collected 339 traces for the unprotected design and 10,000 traces for the protected design.

We use the popular MNIST [5] image classification dataset as a benchmark for our attack method. In the MNIST dataset with 10 classes, we denote the pixel representation of an input image in vector form as $X \in R^{784}$. For algorithmically Hamming distance analysis, we used MATLAB and extracted possible bytes for further analysis with power information extracted from the simulation tool mentioned above.

To mimic realistic simulation for obtaining power traces, we also model and simulate the effect of the PDN with the SPICE simulator described in [37]. The PDN is responsible for delivering a stable voltage to each transistor in the system. A typical chip-level PDN consists of passive components such as inductor (L), capacitor (C), and resistor (R). The values of all R, L, and C which are suggested from [7]. The additional measurement noise to enhance the realistic oscilloscope observations is then computationally injected into the simulated traces using MATLAB.

5.2 Security Assessment of Unprotected Design

The plots for power-based vulnerabilities with an 8-bit query are presented in Figure 3. We observe that the power consumption is minimal when there is an exact match. The observation is valid due to the least discharge of the precharged match-line from source

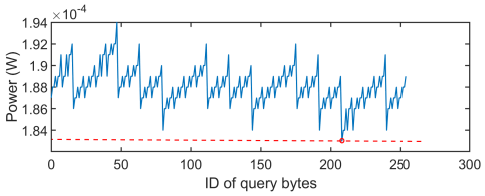


Figure 3: Power variation for all possible candidates of 1-byte query. 0xd0 (208) is considered a class hypervector and it shows the minimum power (in red) with an exact match.

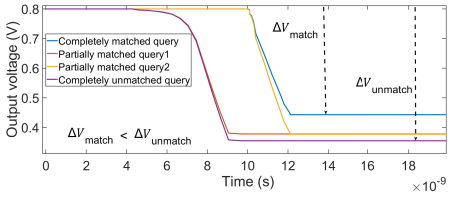


Figure 4: Output voltage discharges least from pre-charged source voltage to stabilized level when there is a match.

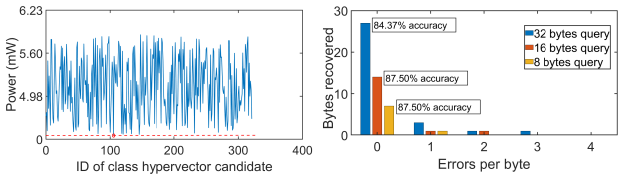
voltage to output voltage (the output voltage level is maximum when there is an exact match). In Figure 4, the voltage difference (ΔV) is minimal when there is an exact match between the query and class hypervectors. With this result, we intuitively make the hypothesis that the query with the lowest power is the actual byte.

To recover the class hypervector, we applied the Algorithm 1. We show the attack results for a 32-byte query in Figure 5a. The minimum power corresponding to the correct candidate is shown with red marker. We achieve a precision of 84.3% in recovering class hypervectors for a 32-byte query, and 87.5% precision for both 16-byte and 8-byte queries, as illustrated in Figure 5b.

In our work, ReRAM is used with an HRS/LRS ratio of 100[2]. Alternatively, other memristive devices like STT-MRAM with an HRS/LRS ratio of ≈3[3], and PCM with an HRS/LRS ratio of ≈450[25], can be employed. The increment of HRS/LRS ratio leads to an approximately linear (the relation is not perfectly linear due to the effect of PDN and measurement noise) increment in power traces as shown in Figure 6a. Therefore, our attack method also works for other memristive devices with more or less traces to analyze the vulnerability. Moreover, in various memristive technologies, it is possible to change the device resistivity through material stack engineering [47]. We investigate whether such a shift in resistivity has an impact on our attack method. Figure 6b shows that power consumption is almost linear with increasing HRS and LRS (keeping the HRS / LRS ratio constant), which shows that higher resistance levels lead to higher vulnerability.

5.3 Security Assessment of Protected Design

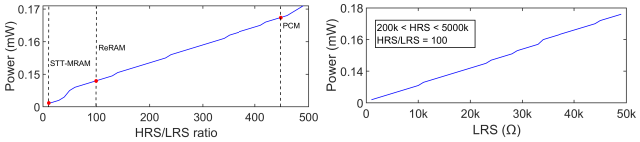
In the protected design, the minimum power consumption does not correspond to the matching query, as shown in Figure 7. In fact, the matched query has a power level (indicated by red circle) in between the maximum and minimum power. We note that the algorithmic attack still allows unveiling candidates for the class hypervector, but side-channel analysis is not possible. Therefore, we confirm that the proposed countermeasure is indeed effective in hiding the



(a) Power analysis on most probable candidates shows that the power is minimum (red marker) for the correct candidate ID.

(b) Number of bytes recovered in each case for 32-byte, 16-byte, and 8-byte query with the number of errors per byte.

Figure 5: Successful power attack with only traces of 338 candidates for class-4 hypervector. First 32 bytes are recovered with high accuracy for class-4 hypervectors in MNIST.



(a) Power with increasing HRS/LRS ratios.

(b) Power with increasing LRS (HRS/LRS ratio fixed).

Figure 6: The effect of different HRS and LRS on power consumption and attack vulnerability.

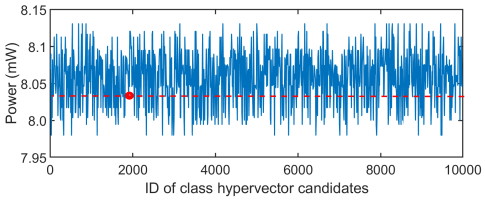


Figure 7: Unsuccessful side-channel collision attack shows the correct candidate is indistinguishable in protected design.

data-dependent power consumption of the CiM-array. However, this comes at the cost of significantly high power (≈ 87% increase) and area overhead (≈ 74% increase). we will investigate countermeasures with less overhead so that the system's performance and efficiency are not significantly compromised while maintaining a high level of security and robustness.

6 CONCLUSION

In this work, we analyze the security of HDC models w.r.t. recovery of class hypervectors. We introduced a two-step attack method to first reduce the search space of potential hypervector candidates and then determine the correct candidate using a side-channel-based collision attack. We successfully recover MNIST class hypervectors in an HDC setup, implemented in a CAM-based CiM architecture. Our findings suggest that HDC is inherently more vulnerable to model-stealing attacks, both algorithmically as well as using power analysis attacks. Therefore, we propose a countermeasure tailored to CiM architecture based on performing additional computations in the crossbar and prove that it can thwart the proposed attack.

REFERENCES

[1] Alberto Annovi et al. 2017. AM06: the Associative Memory chip for the Fast TracKer in the upgraded ATLAS detector. *Journal of Instrumentation* 12, 04 (2017), C04013.

[2] Christopher Bengel et al. 2020. Variability-aware modeling of filamentary oxide-based bipolar resistive switching cells using SPICE level compact models. *IEEE TCAS-I: Regular Papers* 67, 12 (2020), 4618–4630.

[3] Fabrice Bernard-Granger et al. 2015. SPITT: A magnetic tunnel junction SPICE compact model for STT-MRAM. In *DATE*.

[4] Wencheng Chen et al. 2021. Adversarial attacks on voice recognition based on hyper dimensional computing. *Journal of Signal Processing Systems* 93 (2021), 709–718.

[5] Li Deng. 2012. The mnist database of handwritten digit images for machine learning research [best of the web]. *IEEE signal processing magazine* 29, 6 (2012), 141–142.

[6] Shijin Duan et al. 2022. Hdlock: Exploiting privileged encoding to protect hyperdimensional computing models against ip stealing. In *Proceedings of the 59th ACM/IEEE DAC*. 679–684.

[7] Loke Yip Foo. 2019. Power Delivery Network Step Load Response and 1st Droop Formulation. In *EDAPS*. https://doi.org/10.1109/EDAPS47854.2019.9011667

[8] Pierre-Alain Fouque and Frédéric Valette. 2003. The doubling attack–why upwards is better than downwards. In *Cryptographic Hardware and Embedded Systems-CHES 2003: 5th International Workshop, Cologne, Germany, September 8–10, 2003. Proceedings 5*. Springer, 269–280.

[9] Paweł Gawrychowski et al. 2021. Matching patterns with variables under hamming distance. *arXiv preprint arXiv:2106.06249* (2021).

[10] Lulu Ge and Keshab K Parhi. 2020. Classification using hyperdimensional computing: A review. *IEEE Circuits and Systems Magazine* 20, 2 (2020), 30–47.

[11] Gabriel Goller and Georg Sigl. 2015. Side channel attacks on smartphones and embedded devices using standard radio equipment. In *International Workshop on Constructive Side-Channel Analysis and Secure Design*. Springer, 255–270.

[12] Chuan Guo et al. 2019. Simple black-box adversarial attacks. In *International Conference on Machine Learning*. PMLR, 2484–2493.

[13] Yasmin Halawani et al. 2021. RRAM-based CAM combined with time-domain circuits for hyperdimensional computing. *Scientific reports* 11, 1 (2021), 19848.

[14] Xiaobo Sharon Hu, Michael Niemier, Arman Kazemi, Ann Franchesca Laguna, Kai Ni, Ramin Rajaei, Mohammad Mehdi Sharifi, and Xunzhao Yin. 2021. In-memory computing with associative memories: A cross-layer perspective. In *2021 IEEE International Electron Devices Meeting (IEDM)*. IEEE, 25–2.

[15] Mohsen Imani et al. 2017. Exploring hyperdimensional associative memory. In *2017 IEEE International Symposium on High Performance Computer Architecture (HPCA)*. IEEE, 445–456.

[16] Mohsen Imani et al. 2019. Quanthd: A quantization framework for hyperdimensional computing. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 39, 10 (2019), 2268–2278.

[17] Mohsen Imani et al. 2019. Searchd: A memory-centric hyperdimensional computing with stochastic training. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 39, 10 (2019), 2422–2433.

[18] Chuan-Jia Jhang et al. 2021. Challenges and trends of SRAM-based computing-in-memory for AI edge devices. *IEEE Transactions on Circuits and Systems I: Regular Papers* 68, 5 (2021), 1773–1786.

[19] Pentti Kanerva. 2009. Hyperdimensional computing: An introduction to computing in distributed representation with high-dimensional random vectors. *Cognitive computation* 1, 2 (2009), 139–159.

[20] Robert Karam et al. 2015. Emerging trends in design and applications of memory-based computing and content-addressable memories. *Proc. IEEE* 103, 8 (2015), 1311–1330.

[21] Geethan Karunaratne et al. 2020. In-memory hyperdimensional computing. *Nature Electronics* 3, 6 (2020), 327–337.

[22] Arman Kazemi et al. 2021. Fefet multi-bit content-addressable memories for in-memory nearest neighbor search. *IEEE Trans. Comput.* 71, 10 (2021), 2565–2576.

[23] Arman Kazemi et al. 2021. Mimhd: Accurate and efficient hyperdimensional inference using multi-bit in-memory computing. In *2021 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)*. IEEE, 1–6.

[24] Denis Kleyko et al. 2016. Holographic graph neuron: A bioinspired architecture for pattern processing. *IEEE transactions on neural networks and learning systems* 28, 6 (2016), 1250–1262.

[25] Manuel Le Gallo and Abu Sebastian. 2020. An overview of phase-change memory device physics. *Journal of Physics D: Applied Physics* 21 (2020).

[26] Jing Li et al. 2013. 1 mb 0.41 μm^2 2t-2r cell nonvolatile tcam with two-bit encoding and clocked self-referenced sensing. *IEEE Journal of Solid-State Circuits* 49, 4 (2013), 896–907.

[27] Chi-Sheng Lin et al. 2003. A low-power precomputation-based fully parallel content-addressable memory. *IEEE Journal of Solid-State Circuits* 38, 4 (2003), 654–662.

[28] Che-Kai Liu et al. 2022. Cosime: Fefet based associative memory for in-memory cosine similarity search. In *Proceedings of the 41st IEEE/ACM International Conference on Computer-Aided Design*. 1–9.

[29] Dongning Ma et al. 2024. Hyperdimensional computing vs. neural networks: Comparing architecture and learning process. In *2024 25th ISQED*. IEEE, 1–5.

[30] Peter L Montgomery. 1987. Speeding the Pollard and elliptic curve methods of factorization. *Mathematics of computation* 48, 177 (1987), 243–264.

[31] M Niemier et al. 2023. Cross Layer Design for the Predictive Assessment of Technology-Enabled Architectures. In *2023 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 1–10.

[32] Kostas Pagiamtzis et al. 2006. Content-addressable memory (CAM) circuits and architectures: A tutorial and survey. 41, 3 (2006), 712–727.

[33] Abbas Rahimi et al. 2016. Hyperdimensional biosignal processing: A case study for EMG-based hand gesture recognition. In *2016 IEEE International Conference on Rebooting Computing (ICRC)*. IEEE, 1–8.

[34] Abbas Rahimi et al. 2016. A robust and energy-efficient classifier using brain-inspired hyperdimensional computing. In *Proceedings of the 2016 international symposium on low power electronics and design*. 64–69.

[35] Abbas Rahimi et al. 2017. High-dimensional computing as a nanoscalable paradigm. *IEEE Transactions on Circuits and Systems I: Regular Papers* 64, 9 (2017), 2508–2521.

[36] Mark Randolph et al. 2020. Power side-channel attack analysis: A review of 20 years of study for the layman. *Cryptography* 4, 2 (2020), 15.

[37] Brojogopal Sapui et al. 2023. Power Side-Channel Attacks and Countermeasures on Computation-in-Memory Architectures and Technologies. In *2023 ETS*. IEEE, 1–6.

[38] Kenny Schlegel et al. 2022. A comparison of vector symbolic architectures. *Artificial Intelligence Review* 55, 6 (2022), 4523–4555.

[39] Kai Schramm et al. 2003. A new class of collision attacks and its application to DES. In *Fast Software Encryption: 10th International Workshop, FSE 2003, Lund, Sweden, February 24-26, 2003. Revised Papers 10*. Springer, 206–222.

[40] Laura Smets et al. 2023. Training a HyperDimensional Computing Classifier using a Threshold on its Confidence. *arXiv preprint arXiv:2305.19007* (2023).

[41] C-L Sotiropoulou et al. 2017. The associative memory system infrastructures for the ATLAS fast tracker. *IEEE Transactions on Nuclear Science* 64, 6 (2017), 1248–1254.

[42] X Sun et al. 2021. PCM-based analog compute-in-memory: Impact of device non-idealities on inference accuracy. *IEEE Transactions on Electron Devices* 68, 11 (2021), 5585–5591.

[43] Chao Wang et al. 2020. Design of an area-efficient computing in memory platform based on STT-MRAM. *IEEE Transactions on Magnetics* 57, 2 (2020), 1–4.

[44] Stefan Wiefels et al. 2020. HRS Instability in Oxide-Based Bipolar Resistive Switching Cells. *IEEE Transactions on Electron Devices* 10 (2020). https://doi.org/10.1109/TED.2020.3018096

[45] Fangfang Yang et al. 2020. Adversarial attacks on brain-inspired hyperdimensional computing-based classifiers. *arXiv preprint arXiv:2006.05594* (2020).

[46] Fangfang Yang et al. 2020. On the vulnerability of hyperdimensional computing-based classifiers to adversarial attacks. In *NSS 2020, Melbourne, VIC, Australia, November 25–27, 2020, Proceedings 14*. Springer, 371–387.

[47] J Joshua Yang and Sothers. 2013. Memristive devices for computing. *Nature nanotechnology* 8, 1 (2013), 13–24.

[48] Xiaoya Yang et al. 2021. Near and Far Collision Attack on Masked AES. In *The 10th International Conference on Computer Engineering and Networks*. Springer, 810–817.

[49] Sung-Ming Yen et al. 2006. Relative doubling attack against montgomery ladder. In *Information Security and Cryptology-ICISC 2005: 8th International Conference, Seoul, Korea, December 1-2, 2005, Revised Selected Papers 8*. Springer, 117–128.

[50] Sizhe Zhang et al. 2023. Adversarial Attack on Hyperdimensional Computing-based NLP Applications. In *2023 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 1–6.