arXiv:2508.03866v1 [cs.AR] 5 Aug 2025

FlashVault: Versatile In-NAND Self-Encryption with Zero Area Overhead

Seock-Hwan Noh[1], Hoyeon Lee[1], Junkyum Kim[2], Junsu Im[3], Jay H. Park[4], Sungjin Lee[3], Sam H. Noh[5], Yeseong Kim[1†], Jaeha Kung[6†]
†Corresponding authors

[1]DGIST [2]Georgia Tech [3]POSTECH
[4]Samsung Electronics [5]Virginia Tech [6]Korea University

{nosh3332, lhyzone, yeseongkim}@dgist.ac.kr, jun-kyum.kim@gatech.edu,
{junsuim, sungjin.lee}@postech.ac.kr, jayhpark530@gmail.com, samnnoh@vt.edu, jhkung@korea.ac.kr

***Abstract*—We present FlashVault, an in-NAND self-encryption architecture that embeds a reconfigurable cryptographic engine into the unused silicon area of a state-of-the-art 4D V-NAND structure. FlashVault supports not only block ciphers for data encryption but also public-key and post-quantum algorithms for digital signatures, all within the NAND flash chip. This design enables each NAND chip to operate as a self-contained enclave without incurring area overhead, while eliminating the need for off-chip encryption. We implement FlashVault at the register-transfer level (RTL) and perform place-and-route (P&R) for accurate power/area evaluation. Our analysis shows that the power budget determines the number of cryptographic engines per NAND chip. We integrate this architectural choice into a full-system simulation and evaluate its performance on a wide range of cryptographic algorithms. Our results show that FlashVault consistently outperforms both CPU-based encryption (1.46∼3.45×) and near-core processing architecture (1.02∼2.01×), demonstrating its effectiveness as a secure SSD architecture that meets diverse cryptographic requirements imposed by regulatory standards and enterprise policies.**

I. Introduction

Today, governments and private enterprises leverage personal data to deliver highly customized policies and services [20], [79], [98]. In doing so, they often collect sensitive information closely tied to individual privacy such as financial records and biometric data. While such data serves as a critical resource for enabling personalized services, it also becomes a prime target for attacks, with severe consequences if leaked. For instance, in 2013 and 2017, massive data breaches at two major U.S. corporations, Target (a retailer) and Equifax (a credit reporting agency), respectively, exposed sensitive personal information of millions of individuals [19], [35]. These incidents resulted in credit card fraud and identity theft, with financial damages reaching hundreds of billions of dollars. To mitigate such illegal use and data misuse, many countries including the United States, the European Union, Japan, and South Korea have enacted data protection laws that mandate encryption and appropriate technical safeguards when storing personal data on storage devices [24], [28], [57], [106]. In parallel, the industry has adopted standards like TCG Opal [118] and IEEE 1667 [41] to enhance storage security.

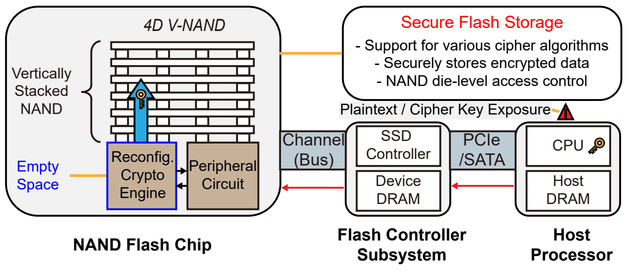

Fig. 1. Overview of FlashVault, which embeds a reconfigurable cryptographic engine beneath the 4D V-NAND to perform data encryption and signing directly within NAND. This in-NAND design enables low-latency encryption and prevents plaintext exposure by avoiding off-chip data movement.

Driven by the increasing adoption of data protection regulations, solid-state drives (SSDs) have adopted the self-encrypting drive (SED) feature to provide hardware-based data protection. SEDs encrypt data using dedicated hardware engines embedded in the device and offer data integrity and access control through digital signatures and authentication mechanisms. However, due to the following limitations, they fall short of addressing evolving security requirements.

[Limitation 1] *Modern SSDs support only a limited set of cryptographic algorithms.* In accordance with data protection laws, many countries require or recommend that public institutions and private enterprises handling personal information adopt national standard cryptographic algorithms [24], [28], [57], [106]. However, current commercial SSDs currently support only specific algorithms, e.g., AES, SHA, and RSA [34], [70], [94], [102]. As a result, whenever unsupported algorithms are required, encryption is offloaded to the software stack on the host CPU [107]. This software-based encryption incurs I/O latency and may increase system-wide response time [16]. Such delays not only degrade the user experience in public and enterprise services but also cause significant disruption in latency-sensitive applications, such as high-frequency trading systems [71] and decentralized exchanges [114].

[Limitation 2] *Unencrypted data may be stored in NAND flash memory.* Modern SSDs are equipped with data encryption and SED features [68], [117]. However, the activation of these

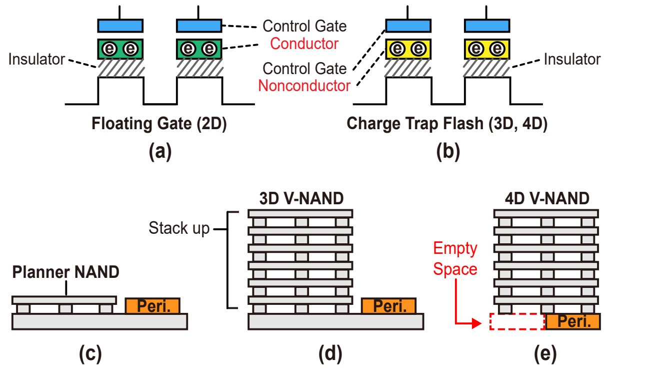

Fig. 2. Device structures of (a) a floating gate transistor and (b) a charge trap flash. Structures of (c) 2D, (d) 3D and (e) 4D NAND flash memories.

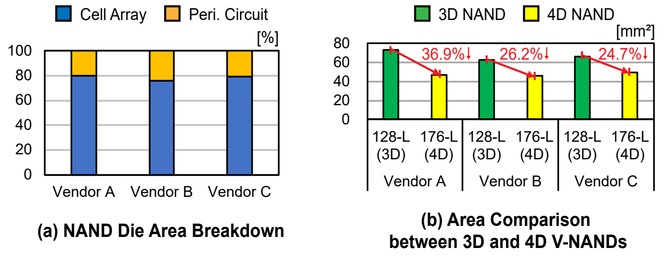

Fig. 3. (a) Area breakdown of a 512 Gb 3D V-NAND chip into the cell array and peripheral circuits for three major vendors. Peripheral circuits include components such as row decoders, page buffers, and charge pumps. (b) Comparison of die sizes between 128-layer 3D NAND and 176-layer 4D NAND flash from the same vendors. Sources: [10], [18], [43], [86], [103].

features is typically determined by user configuration [27], [112]. Thus, if encryption is not enabled, unencrypted data (i.e., plaintext) is stored in NAND flash memory. Even when encryption is enabled, residual traces of cryptographic keys may remain in NAND flash due to garbage collection (GC) and wear-leveling (WL) during the key revocation process [3], [121]. In addition, if a power loss occurs, plaintext data residing in the DRAM of the SSD may be flushed to NAND flash [2], [128].

[Limitation 3] *NAND flash lacks architectural protections against software-based and physical attacks.* Since commercial SSDs do not support trusted execution environments within their NAND flash chips, attackers can manipulate the flash translation layer (FTL) mapping tables to distort the data organization or analyze FTL logs to retrieve or tamper with residual data that have been logically deleted but remain physically present [47]. In addition, attackers can connect NAND readers to the flash chip to extract unencrypted data, or remnants of data left behind due to GC and WL [68]. Beyond these device-level threats, when encryption operations are offloaded to the host CPU, the risk of exposure to attacks increases further, weakening overall system security. For instance, plaintext data transmitted between the SSD and host may be intercepted through bus probing [122], [123], and cryptographic keys may be inferred or stolen through power analysis [90], [91] or electromagnetic analysis [30], [97] during CPU-based encryption processes.

To address these limitations of existing SSDs, we propose *FlashVault*, the first in-NAND self-encryption architecture that supports a wide range of cryptographic algorithms directly within NAND flash (Fig. 1). It is specifically designed for modern 4D V-NAND architectures, which adopt a peripheral-under-cell structure with unused space beneath the memory array. By embedding reconfigurable cryptographic engines into this unused space, FlashVault enables a fast and secure in-NAND enclave without incurring additional area overhead.

The specific architectural features of FlashVault are:

1) **Reconfigurable cryptographic engine:** FlashVault integrates a reconfigurable cryptographic engine that supports a variety of block ciphers and public-key cryptographic algorithms (PKC). It also supports post-quantum cryptography (PQC), providing the architectural flexibility needed to address emerging security threats posed by quantum computing.
2) **Utilization of the unused area beneath 4D V-NAND:** In FlashVault, the cryptographic engine is placed in the unused silicon area beneath the 4D V-NAND array, enabling secure functionality without incurring additional area overhead. By performing encryption and decryption near the memory array, FlashVault ensures data confidentiality and integrity at the NAND chip level. Furthermore, by internalizing operations that were traditionally offloaded to the CPU, FlashVault eliminates potential attack surfaces along the data path to the host.
3) **Die-level shared integration architecture:** FlashVault supports high-throughput encryption and decryption by employing a pipelined, die-level shared integration of cryptographic engines within each NAND chip. This architecture facilitates the deployment of computationally intensive cryptographic algorithms that were previously impractical for real-time secure applications.

This paper focuses on the architecture of FlashVault and its performance. While security functionality is a fundamental requirement driven by legal and industrial standards, performance is also a critical factor for the practical deployment of secure storage systems in real-world environments [17], [47], [58]. Moreover, the structural characteristics of FlashVault inherently mitigate security vulnerabilities observed in conventional systems (Section II-C).

II. Preliminaries

A. NAND Flash Memory Technology

The memory cells in early-generation SSDs are based on floating gate transistors (FGTs). An FGT is a complementary metal-oxide-semiconductor (CMOS) technology capable of holding electrical charges in an isolated silicon conductive layer (Fig. 2-(a)) [127]. Early generation flash memory arrays have a 2D planar structure with FGTs placed side-by-side on the die (Fig. 2-(c)). However, as the process node has scaled down, FGTs encountered fabrication challenges, i.e., charge leakage caused by the thin conductive layer and intensified cell-to-cell interference [12], [60]. To alleviate this limitation, FGTs were gradually superseded by transistors utilizing

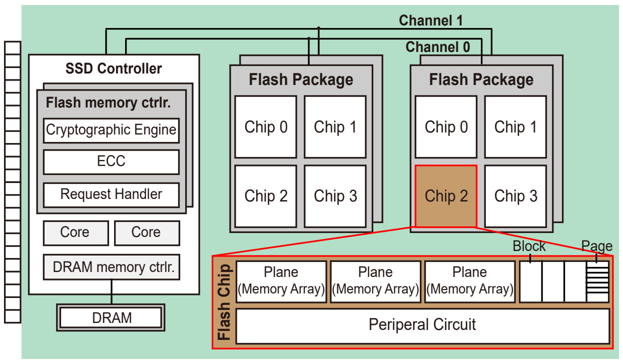

Fig. 4. Internal architecture of modern SSDs.

charge trap technology. These newer transistors employ a non-conductive silicon layer as a control gate, highly effective at retaining electrical charges (Fig. 2-(b)) [12], [127]. Moreover, to meet demands for higher density, memory suppliers developed a 3D NAND structure (Fig. 2-(d)), featuring vertically stacked charge trap transistors alongside the peripheral circuit on the same integrated chip [15], [87], [113]. This 3D architecture significantly increased storage capacity by adding more layers achieving lower cost/bit. More recently, the industry has introduced 4D NAND flash products[1], which adopt an advanced vertical integration scheme to improve bit density (Fig. 2-(e)). These offer a reduced chip area compared to 3D NAND flash by positioning the peripheral circuit underneath the memory array [51]. Fig. 3-(a) illustrates the area composition of 512 Gb-class 3D V-NAND chips from three major memory manufacturers, indicating that peripheral circuits account for approximately 20∼24% of the total chip area. By placing the peripheral circuits under the memory array, the 4D structure not only reduces the die footprint by a comparable amount (Fig. 3-(b)), but also leaves a portion of the bottom silicon layer unoccupied, since the relocated circuits are smaller in area than the memory array above. Motivated by this architectural characteristic, there have been efforts to leverage the resulting unused area by integrating auxiliary logic circuits, such as pattern matchers [18], data processing units [108], and neuron-inspired circuits [39]. This work leverages this area to embed reconfigurable cryptographic engines, enabling fast and secure near-data cryptographic processing.

B. Internal Architecture of SSDs

Fig. 4 illustrates the typical architecture of modern SSDs [12], [32], [48], [115]. An SSD consists of NAND flash packages, an SSD controller, and DRAM. Each NAND flash package contains multiple flash chips, each housing several memory planes. These planes are further divided into blocks and pages. A block is the smallest erasable unit, typically holding 1,024 pages. A page, the basic unit for read and program/write operations, usually has a size between 4 KB and 16 KB. Data read from or programmed to the flash memory is managed by a flash memory controller. This controller orchestrates data distribution to prevent bottlenecks among NAND packages sharing a byte-wide channel. It also performs error correction on data read from the memory array and performs encryption and decryption of specific cipher algorithms, i.e., AES, SHA, and RSA, via a dedicated cryptographic engine. The SSD controller, comprising 1 to 4 flash memory controllers, is equipped with an embedded multicore processor. This processor operates the internal firmware known as the flash translation layer (FTL). The FTL is responsible for translating addresses between the logical memory space and the physical flash page, performing garbage collection (GC) to reclaim invalid or empty spaces, and executing wear leveling (WL) to ensure even memory usage across the memory blocks. The DRAM is utilized for storing a virtual-to-physical address mapping table and the firmware code of the FTL.

[1]In the industry, 4D V-NAND is also referred to as PUC (Periphery Under Cell) by SK hynix, CUA (CMOS Under Array) by Micron, and COP (Cell Over Periphery) by Samsung [43].

C. Threat Model

Modern commercial SSDs provide hardware-based data protection through SED feature, typically employing AES encryption with a dedicated engine in the SSD controller [68], [117]. These controllers also support cryptographic primitives such as SHA and RSA for integrity verification and firmware authentication [34], [70], [94], [102]. In current SSD architectures, encryption is typically performed in one of two ways: (i) by a hardware engine integrated into the SSD controller or (ii) in software on the host CPU when the required algorithm is not supported by the controller.

When the *SSD controller* performs encryption, the NAND flash stores encrypted data sent from the controller [47]. However, storing encrypted data in NAND does not eliminate all security risks. For example, remnants of encryption keys or plaintext data may persist in NAND flash due to WL, GC [3], [121], or unexpected power loss [2], [128]. These residual traces may be extracted through physical attacks using flash readers [68]. Moreover, off-the-shelf NAND chips lack built-in access control mechanisms, leaving them vulnerable to unauthorized access through malicious firmware [68].

When encryption is performed by the *CPU*, sensitive data can be exposed to adversaries through probing attacks on the data bus [122], [123]. Attackers may also analyze power consumption patterns [90], [91] or electromagnetic emissions [30], [97] using sensors or oscilloscopes to infer secret keys or other sensitive information. Furthermore, software-based encryption is at risk of microarchitectural attacks, such as cache timing side channels, which can leak sensitive intermediate results that depend on cryptographic secrets [83], [124].

FlashVault mitigates these threat models by placing dedicated cryptographic engines beneath the memory array of 4D V-NAND. These engines are physically located in the deep silicon layer underneath the stacked array, effectively removing externally accessible surfaces and preventing physical access-based attacks. FlashVault ensures that only encrypted data remains in NAND even under conditions such as WL, GC, or power loss through in-NAND encryption. When used in conjunction with the encryption built into the SSD controller, FlashVault creates a secure enclave that spans the entire data path from the controller to the NAND flash media. Furthermore, FlashVault supports PKC and PQC directly within the NAND layer, providing on-die authentication to block unauthorized firmware accesses. Finally, by eliminating the need to offload cryptographic workloads to the host CPU, FlashVault inherently avoids a wide range of CPU-side vulnerabilities, including software attacks and side-channel leakage.

III. Motivation

A. Support for Diverse Cryptographic Algorithms

1) Block Cipher Algorithm: Modern SSDs provide a strong security level by employing block cipher algorithms that enable fast encryption and decryption to protect data. Most commercial self-encrypting SSDs support fixed block cipher algorithms such as AES [42], [44], [52], following storage security standards such as FIPS 140-2/3 [73], [74], TCG Opal [118], and IEEE 1619 [42]. However, modern security requirements are becoming increasingly diverse. To protect data vital to national security and public welfare, and maintain cryptographic sovereignty, many countries have enacted stricter data protection laws. Accordingly, they mandate or recommend the use of national standard block cipher algorithms to protect sensitive data (e.g., personal identifiers, financial records, and military information) stored in storage media. For example, the United States, the European Union, South Korea, China and Russia require the use of AES/Triple DES [77], AES [28], ARIA/SEED/HIGHT [56], SM4 [33], and GOST [119], respectively. Japan and Turkey recommend Camellia [23] and KET [53] for securing public sector data. This growing diversity of security demands indicates that relying on a single fixed block cipher algorithm in SSDs is inadequate to address today's heterogeneous requirements.

2) Public-Key Cryptographic Algorithm: Recent storage devices are equipped with advanced security features based on data integrity and digital signatures. Storage security standards such as TCG Opal [118] and NVMe-oF specify public-key-based authentication mechanisms as core components, leading modern SSDs to support a limited set of public-key cryptographic algorithms such as RSA and NIST ECC-crypto[2] (e.g., secp256r1) [34], [70], [94], [102]. However, these fixed-function designs inherently lack flexibility to accommodate diverse regional security requirements. For instance, Germany's Federal Office for Information Security recommends ECC-crypto based on Brainpool curves [29], while China promotes its domestic SM2 algorithm in industry [105]. However, commercial SSDs generally do not support such region-specific cryptographic algorithms, limiting their applicability in regions with specific regulatory requirements.

Moreover, as storage devices are entrusted with preserving sensitive data over long periods, ensuring their long-term security against cryptographic threats is critical. From this perspective, the advent of quantum computing poses a threat to existing public-key cryptosystems, since quantum computers can solve the underlying mathematical problems such as integer factorization and discrete logarithms using Shor's algorithm [101]. This raises the concern that today's widely adopted public-key security mechanisms may become ineffective in the near future. To address this, NIST has led post-quantum standardization and selected Dilithium, FALCON, and SPHINCS+ as digital signature standards [75]. In response, supporting these PQC algorithms is emerging as a critical requirement for future storage systems that can withstand future cryptographic threats.

[2]ECC-crypto refers to Elliptic Curve Cryptography.

B. Limits of CPU-based Encryption

Modern SSDs have evolved beyond simple data storage devices into intelligent security platforms equipped with various built-in protection mechanisms. In addition to encrypting data to ensure confidentiality, these devices also support integrity-focused features such as secure boot and tamper-proof logging, which perform signature verification and detect unauthorized modifications during system initialization and log management. To implement these security functions effectively, different cryptographic algorithms may be selected depending on the application domain and security priorities. For example, SSDs deployed in security-critical sectors, such as defense, may employ robust signature schemes such as ECDSA [80] or upcoming post-quantum algorithms to ensure high assurance. On the other hand, SSDs in latency-critical environments like financial trading systems may use lightweight block ciphers such as HIGHT [37] or Camellia [4] to minimize encryption latency while still ensuring data protection.

However, off-the-shelf SSDs typically support only a limited set of algorithms at the hardware level [34], [70], [94], [102], requiring other cryptographic operations to be offloaded to the host processor [107]. This CPU-based processing often incurs significant latency, making it difficult to meet the stringent timing requirements of practical applications. To mitigate this issue, it is necessary to minimize reliance on host-side processing in order to reduce the latency overhead caused by offloading cryptographic operations to the host processor. FlashVault addresses this challenge by supporting in-NAND self-encryption, enabling high-performance execution of diverse cryptographic algorithms. This section motivates the need for in-NAND processing by analyzing the inefficiencies of CPU-based encryption in real-world scenarios.

1) Experimental Setup: To quantitatively evaluate the performance overhead of CPU-based encryption, we measured the latency of various cryptographic algorithms on a desktop system with an Intel Core i7-13700K (16 cores), 32 GB DDR5-5600 memory, and a 2TB SK Hynix Gold P31 M.2 NVMe SSD (PCIe 3.0 ×4), running Ubuntu 22.04 LTS. Block ciphers and public-key cryptographic algorithms were implemented using the Botan library [11], and PQC algorithms were evaluated using the openssl speed benchmark tool in conjunction with liboqs [81] and OpenSSL [82]. For block ciphers, we used fixed 4 KB input messages, whereas public-key and PQC algorithms were evaluated across a range of input sizes. In particular, the experiments on public-key algorithms

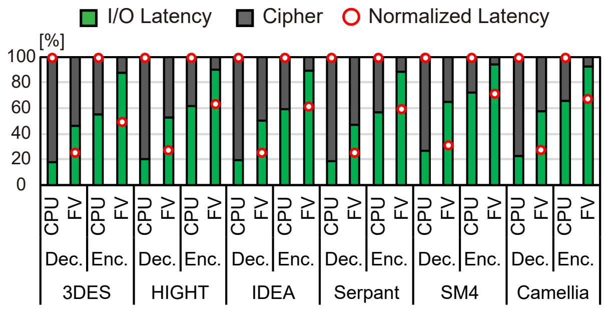

Fig. 5. Normalized latency breakdown of decryption (Dec.) and encryption (Enc.) using six representative block cipher algorithms on CPU and FlashVault (FV). Each bar shows I/O latency (green), cipher latency (gray), with red circles indicating total latency normalized to the CPU implementation.

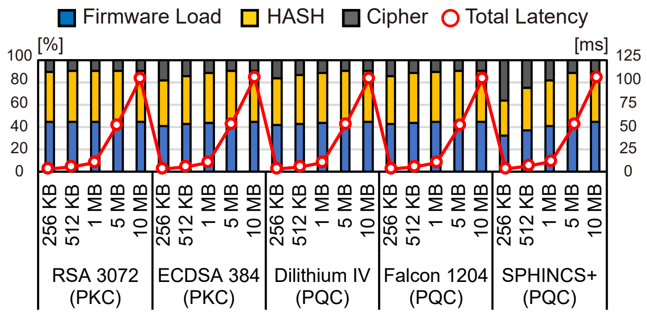

Fig. 6. Latency breakdown of public-key cryptographic verification operations during secure boot, evaluated with various input sizes. The stacked bars show relative contributions of boot image loading (blue), hash computation (orange), and PKC/PQC operations (gray), with red circles indicating the total latency.

were based on scenarios modeling the verification process of secure boot and the signature validation used in security logging systems such as tamper-proof logging. As part of the analysis of secure boot latency, we used the latency value reported in [116] for the boot code loading process. Latency was measured using the `clock_gettime()` function, with `fsync()` and `posix_fadvise()` used to reduce the effects of OS-level file caching. Each algorithm was evaluated 1,000 times per input size, and we report the average latency.

2) Performance of Block Cipher Algorithms: Fig. 5 shows the latency breakdown for six representative block ciphers (3DES [84], HIGHT [37], IDEA [59], Serpent [9], SM4 [33], Camellia [4]) executed in CTR mode, measured under both CPU and FlashVault. Although FlashVault embeds multiple cryptographic engines within each NAND flash chip, Fig. 5 reports the result based on a single-block-per-chip configuration to focus on the impact of where encryption occurs. The experimental setup for FlashVault is described in Section VIII-B. Each bar in the figure is divided into encryption latency and I/O latency to illustrate their respective contributions to the total processing time. In the CPU-based implementation, limited concurrency in cryptographic processing and the frequent data transfers between the CPU and host DRAM contribute significantly to the overall latency. In contrast, when cryptographic operations are performed inside in-NAND flash, as implemented in FlashVault, this latency can be mitigated by eliminating the host-side cryptographic processing. Consequently, FlashVault achieves an average latency reduction of 64.2% for decryption and 37.0% for encryption compared to CPU-based processing.

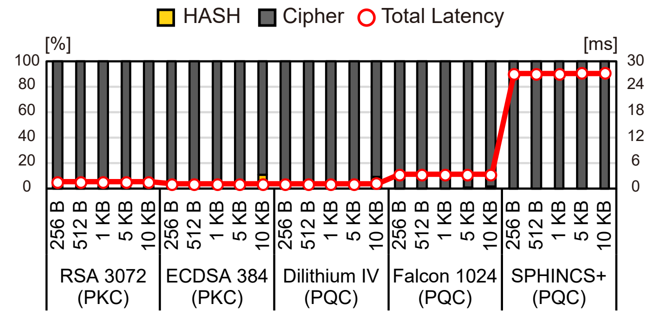

Fig. 7. Latency breakdown of public-key cryptographic signing operations in tamper-proof logging evaluated across various input sizes. Each stacked bar shows the contributions of hash computation (orange) and PKC/PQC operations (gray), and red circles indicate the total latency.

3) Latency of PKC and PQC Verification: Fig. 6 shows the signature verification latency using well-known PKC and PQC during secure boot for boot images from 256 KB to 10 MB. The verification sequence starts by loading the firmware image from boot ROM or NAND flash, computes a cryptographic hash, e.g., SHA [76], and verifies the resulting digest using a digital signature algorithm, e.g., PKC or PQC. The input to the signature verification is relatively small, as it is a fixed-length digest produced by the hash function. However, since boot images range from hundreds of KB to tens of MB, image loading and hash computation account for a large portion of the total verification latency. Meanwhile, the overall verification latency varies with the size of the boot image, increasing as the image size grows. As shown in the figure, when a 10 MB image is used, the signature verification time exceeds 100 ms across all cryptographic algorithms [45]. This result indicates that it is difficult to meet the constraint imposed by commercial hardware implementations, which require boot-time verification to be completed within 100 ms after system power stabilization (i.e., `power_good`). Moreover, as modern high-end SSDs often adopt firmware images larger than 10 MB [95], meeting the timing constraint using CPU-based verification becomes even more challenging.

4) Latency of PKC and PQC Signing: Fig. 7 presents the latency breakdown of digital signature generation using various PKC and PQC algorithms in a tamper-proof logging scenario, where signatures are generated at runtime for individual log entries. Since logging operations typically target small messages ranging from a few hundred bytes to several kilobytes [1], the evaluation was conducted with input message sizes ranging from 256 B to 10 KB. Given such small message sizes, PKC and PQC operations dominate the total latency. Commercial tamper-proof logging systems must complete log generation and signing for each event within 2 to 15 ms to meet real-time processing requirements [1], [99]. However, as shown in the figure, Falcon 1024 and SPHINCS+ require considerable time for signature generation regardless of the input message size. This indicates that several NIST-standardized

TABLE I
PRIMITIVE OPERATIONS OF REPRESENTATIVE CRYPTOGRAPHIC ALGORITHMS

Category	**Algorithm**	**# of Rounds**[*]	**Key Length [bit]**	**Input Unit Size**[†] **[bit]**	**Processing Granularity**[‡] **[bit]**	**Required Primitive Operations**
Block Cipher	AES [96]	10 / 12 / 14	128 / 192 / 256	128	8	XOR, S-Box, Shift, Multiplication
	3DES [84]	48	112 / 168	64	4	XOR, S-Box, Permutation
	IDEA [59]	8.5	128	64	16	XOR, ModAdd, ModMult
	Serpant [9]	32	128 / 192 / 256	128	4	XOR, S-Box, Shift, Permutation
	HIGHT [37]	32	128	64	8	XOR, Shift, ModAdd, ModMult
	SM4 [33]	32	128	128	8	XOR, S-Box, Shift
	Camellia [4]	18 / 24	128 / 192 / 256	128	8	XOR, S-Box, Shift, AND, OR
HASH	SHA-2 [76]	64	-	512	32	XOR, NOT, AND, Shift, ModAdd
PUK	RSA [92]	1	1024 / 2048 / 3072 / 4096	256 / 512	$\geq$ 1024	AND, OR, Shift, ModMult, ModExp
	ECDSA [80]	1	160 / 224 / 256 / 384 / 521	256 / 512	$\geq$ 256	AND, OR, Shift, ModMult, ModInv
PQC	Dilithium [7], [26]	1	2048 / 3072	256 / 512	23	XOR, AND, Shift, NTT, Keccak, ModMult, Comparison
	Falcon [89]	1	896 / 1280	256 / 512	64	XOR, AND, Shift, FFT, ModMult, Comparison
	SPHINCS+ [8]	1	256 / 384 / 512	256 / 512	32	XOR, AND, Shift, Keccak, Comparison

[*] **Round** In PKC and PQC, the overall operation is not performed in a repetitive round-based manner, but rather consists of a single large mathematical operation, which is why the round count is set to 1. However, despite having a round count of 1, the internal operations may involve multiple rounds. For instance, in RSA, the modular exponentiation is performed $O(\log(e))$ times, depending on the length of the public key exponent e [92], whereas in PQC, the Keccak-f1600 permutation function is executed 24 times [63].
[†] **Input Unit Size** In block ciphers and hash functions, the input unit size represents the block size processed in each operation. For public-key cryptography (PUK) and post-quantum cryptography (PQC), it corresponds to the length of the message digest, computed via widely used cryptographic hash functions such as SHA-256 or SHA-512.
[‡] **Processing Granularity** In PUK schemes, the processing granularity represents the bit length of the prime modulus, whereas in PQC, it corresponds to the bit width of polynomial coefficients. Although PQC generally operates on smaller bit-width units than PUK, it achieves comparable security levels through high-degree polynomial operations or hash-based constructions [78].

PQC algorithms are impractical for latency-critical tamper-proof logging when executed on the host CPU, as is typical in modern SSDs.

IV. DESIGN CONSIDERATIONS

This section presents the key design considerations behind FlashVault, an in-NAND self-encryption architecture designed to support a wide range of cryptographic algorithms with low-latency execution.

A. Available Silicon Area

A key feature of FlashVault is integrating cryptographic engines into the unused area under the memory array of a 4D V-NAND chip. To enable this integration in the empty space, we first estimate the available area in the 4D V-NAND. Due to lack of public data, we analyze Samsung's first 4D V-NAND product, the 7th-generation 4D V-NAND chip [14]. We calculate the available space as $M - P$, where M and P are the memory array and peripheral circuit areas. To estimate the peripheral circuit area, we refer to the specifications of Samsung's 6th-generation 3D V-NAND solution [46]. The 7th-generation NAND flash chip improves area efficiency by repositioning the peripheral circuit underneath the memory array. As documented in [18], [31], [38], the peripheral circuit lies entirely under the memory array. Peripheral area is estimated as $D_6 - D_7$, where D_6 and D_7 are die sizes of the 6th and 7th-genration chips. Consequently, we identify a free space of 20.60 mm^2 in the 7th-generation 4D V-NAND flash. **(Takeway 1)** *We leverage this unused area (20.60 mm^2) to integrate cryptographic engines supporting diverse algorithms.*

B. Power Budget

Since a NAND flash chip receives only a limited amount of power from the main power management IC of the SSD [126], any cryptographic engine integrated into the unused space of 4D V-NAND must stay within this power budget. If the

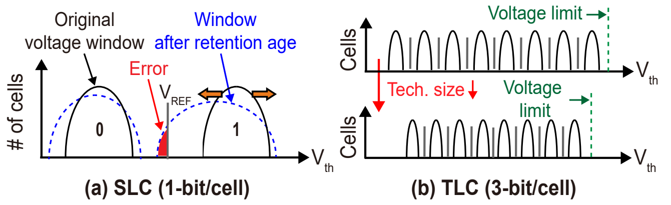

Fig. 8. Threshold voltage (V_{th}) distributions of (a) single-level cell (SLC) and (b) triple-level cell (TLC).

proposed architecture exceeds this budget, it may degrade cell reliability or performance due to thermal issues [64], [125]. Thus, we identify the power budget and ensure that cryptographic engines operate safely within this limit. However, as Samsung's 7th-gen V-NAND power budget is undisclosed, we estimate it based on the power budget of the 6th-generation and the reported efficiency improvements in the 7th-generation. Since block-level programming typically consumes the highest power among NAND flash operations [18], [72], we use it as the basis for power budget estimation. For 6th-generation V-NAND, the programming power can be calculated as follows:

$$P_{\text{program}} = I_{\text{program}} \cdot V_{\max}, \tag{1}$$

where P_{program} denotes the programming power, I_{program} is the programming current, and $V_{\max}$ is the maximum operating voltage. Based on technical documents [46], [93], the programming current and maximum operating voltage of the 6th-generation V-NAND are 13.8 mA and 3.6 V, respectively, resulting in a power budget of 49.68 mW. The 7th-generation V-NAND achieves 16% improvement in power efficiency over the 6th generation [104]; thus, its power budget is estimated to be 41.73 mW. **(Takeway 2)** Given this estimated budget, *we design the cryptographic engines to run in parallel under the power constraint (41.73 mW) within the 4D V-NAND die.*

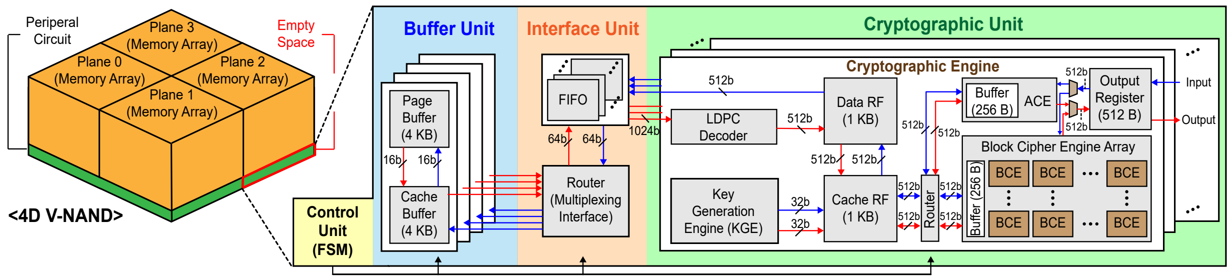

Fig. 9. Overview of the FlashVault architecture. Red and blue arrows indicate the dataflows for read and program operations, respectively. BCE and ACE represent the block cipher engine and the asymmetric cipher engine (supporting PKC and PQC).

C. Reconfigurable Cryptographic Engine

One key design goal of FlashVault is supporting diverse cryptographic algorithms through a versatile cryptographic engine. To identify the fundamental cryptographic primitives used across different schemes, we analyzed 17 block ciphers from the Botan library [11]; public-key cryptosystems based on integer factorization and elliptic curve problems; and PQC algorithms, including lattice-based and hash-based schemes, from the liboqs library [81]. Table I summarizes the key primitive operations used in widely adopted block ciphers and public-key algorithms, and NIST-standardized PQC schemes [75]. Our analysis reveals that block ciphers are typically composed of five basic operations: modular arithmetic, logical operations, permutation, substitution (S-Box), and bitwise shift. PKC and PQC algorithms, on the other hand, primarily rely on logical operations, bitwise shifts, number-theoretic transforms, Keccak, modular arithmetic, and comparison operations. **(Takeway 3)** Based on these observations, *we develop a reconfigurable cryptographic engine that supports diverse primitives, enabling FlashVault to efficiently implement various algorithms within the NAND die.*

D. Error Correction

NAND technology scaling has evolved beyond physical scaling, transitioning from single-level cell (SLC) to triple-level cell (TLC) designs. While multi-bit memory cells significantly increase storage density and reduce cost per bit, this transformation has deteriorated the reliability characteristics of NAND flash, particularly in terms of data retention and endurance. Fig. 8-(a) shows the threshold voltage (V_{th}) distributions of SLC and how they are affected by various error sources. According to previous studies [50], [64], [65], [88], [100], reading or programming a flash cell slightly increases the V_{th} of adjacent cells by unintentionally injecting electrons into their charge traps (i.e., cell-to-cell interference). In addition, the leakage of electron charges over time decreases the V_{th} level (i.e., retention loss). As a result of these mechanisms, the V_{th} level can drift across the reference voltage (V_{REF}), leading to bit errors. As shown in Fig. 8-(b), TLC has narrower voltage windows for each state than SLC, and the technology scaling further shrinks the size of these windows. To ensure data reliability under such conditions, modern SSDs incorporate error correction codes (ECC) within their controllers. Since FlashVault performs cryptographic operations directly within the NAND chip, it is essential to correct bit errors in advance to ensure only error-free data is encrypted. **(Takeway 4)** To support reliable in-NAND self-encryption, *FlashVault integrates error correction capabilities into the NAND flash chip, allowing correction to take place on-chip.*

V. FLASHVAULT ARCHITECTURE

In this section, we present an overview of the proposed on-die self-encryption architecture, FlashVault. FlashVault integrates cryptographic engines into the unused area of a NAND flash chip. This strategic integration utilizes idle silicon, distinguishing FlashVault from state-of-the-art SSDs. Fig. 9 illustrates the detailed architecture of FlashVault, comprising (i) a control unit, (ii) a buffer unit, (iii) an interface unit, and (iv) a cryptographic unit embedded in the unused die area.

1) Control Unit: FlashVault includes a control unit based on a finite-state machine (FSM), which operates as a supplementary unit to the SSD controller. It receives control signals issued by the SSD controller firmware in response to host I/O commands. Based on these signals, the FSM manages the control flow and selects the appropriate hardware resources and dataflow paths for each state.

2) Buffer Unit: The buffer unit comprises a 4 KB page and cache buffers. The page buffer holds data directly sensed from the vertically stacked memory array, while the cache buffer mirrors this data. This double-buffering scheme enhances I/O performance by allowing the cache buffer to forward previously transferred data to downstream modules in parallel with ongoing read or program operations in the page buffer.

3) Interface Unit: The interface unit arbitrates plane access to the cryptographic units through an internal router. It temporarily buffers incoming data from the buffer unit, aligns it, and forwards it to the cryptographic unit via the input bus.

4) Cryptographic Unit: The core component of FlashVault is the cryptographic unit, which consists of multiple cryptographic engines. Each engine is configured as an array of *block cipher engines (BCEs)* for symmetric encryption and an *asymmetric cipher engine (ACE)* supporting PKC and PQC [3]. When FlashVault operates in self-encrypting mode, block ciphers are executed for every I/O request. These algorithms allow subword-level parallelism across multiple processing stages

[3]Block ciphers use the same secret key for both encryption and decryption, whereas PKC and PQC use asymmetric key pairs comprising a public key for encryption and a private key for decryption.

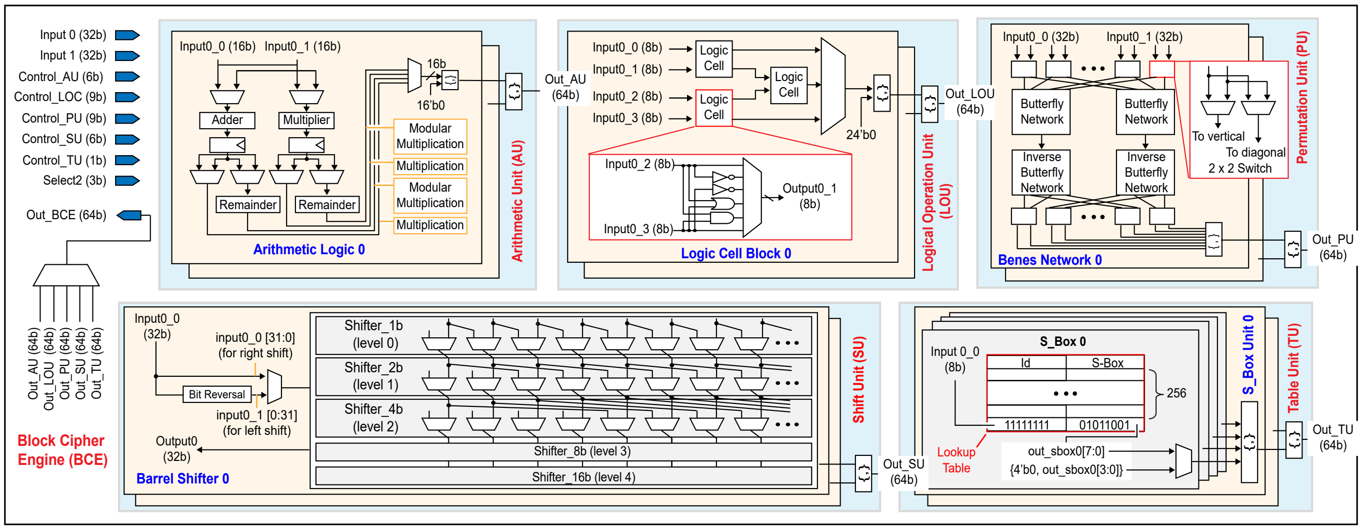

Fig. 10. Schematic diagram of the Block Cipher Engine (BCE). Each BCE has five fundamental building blocks: arithmetic unit (AU), logical operation unit (LOU), permutation unit (PU), shift unit (SU), and table unit (TU).

(i.e., each round in the cipher) [36]. Thus, FlashVault adopts a SIMD-style array of 16 BCEs, enabling high throughput. The ACE module is activated during signature generation and verification. Since most cryptographic operations in PKC and PQC, excluding hash computations, are performed sequentially [55], FlashVault incorporates a single ACE module.

In addition to the cipher engines, the cryptographic engine integrates auxiliary components to support full in-NAND cryptographic functionality. Among them, a *low-density parity-check (LDPC) decoder* performs error correction on data read from the NAND array. Unlike conventional SSDs, which perform error correction in the SSD controller, FlashVault conducts in-situ correction within the NAND flash chip. For this purpose, the LDPC decoder first corrects raw data errors from the NAND array and then forwards the decoded data to two dual-purpose register files: the Data Register File (DRF) and the Cache Register File (CRF). FlashVault adopts the LDPC decoder in [21], which uses a gradient descent bit-flipping algorithm. It initializes all bits (i.e., variable nodes) using a hard decision, which interprets the signal as 0 or 1. The decoder then iteratively flips the bit with the lowest reliability score, determined from the parity check results and NAND read channel conditions. This process continues until all parity constraints are satisfied or a predefined iteration threshold is reached. Despite being based on a hard-decision architecture with a rate-0.88 quasi-cyclic LDPC code, the decoder provides error correction performance comparable to soft-decision decoders that use probabilistic signal interpretation.

As another auxiliary component, FlashVault includes a Physical Unclonable Function (PUF)-based *key management engine (KME)*. This engine integrates a ring oscillator (RO) PUF that leverages process-induced variations in oscillator frequencies to generate a chip-unique bit pattern, which is used as the root key [67]. The root key is then expanded by a key derivation function (KDF) into symmetric session keys and asymmetric key pairs. The KDF uses a secure hash function, combining the root key with additional salt (i.e., a random value) or context, producing keys with varying lengths and formats, tailored to meet security requirements. The derived keys are stored in the CRF and delivered to either the BCEs or the ACE via the internal router.

VI. Cipher Engine Microarchitecture

This section describes the core modules of FlashVault building on the algorithmic analysis discussed in Section IV-C.

A. Block Cipher Engine (BCE)

As discussed in Section IV-C, we identify five fundamental operations commonly used in block cipher algorithms: modular arithmetic, logical, permutation, S-Box, and shift operations. Accordingly, we implement five corresponding compute units, that is, arithmetic, logical operation, permutation, shift, and table, in the BCE. These units are reconfigurable to support various block cipher operations. Fig. 10 shows the microarchitecture of the BCE, which integrates these five units.

1) Arithmetic Unit (AU): Some block cipher algorithms require modular addition and/or multiplication. To support these operations efficiently, the AU features dedicated hardware components: an adder, a multiplier, and two remainder circuits. As illustrated in Fig. 10, the outputs of the adder and multiplier can be routed either to the remainder circuits for modular operations or directly to the output for non-modular operations.

2) Logical Operation Unit (LOU): Block cipher algorithms use the same key for encryption and decryption, performing these operations through bitwise XOR between the key and input data. The LOU supports XOR and extends to additional operations such as OR, AND, NOT, and their combinations.

3) Permutation Unit (PU): Block cipher algorithms utilize permutation operations to obscure the relationship between the input (i.e., plaintext) and the output (i.e., ciphertext) during encryption. The PU is equipped with two 32-bit Benes networks to support this. These networks can either be combined to

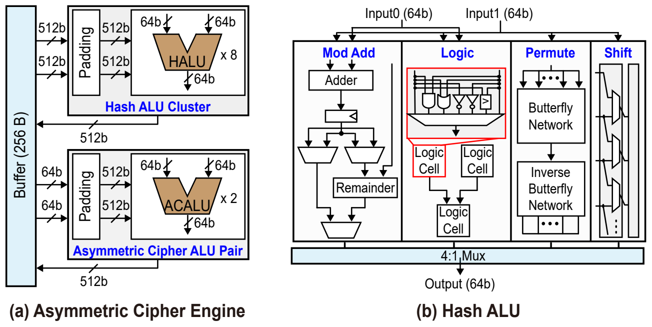

Fig. 11. (a) Architectural schematic of the asymmetric cryptographic engine (ACE), consisting of a hash ALU cluster and a pair of asymmetric cipher ALUs. (b) Internal structure of the hash ALU.

produce a single 64-bit permutation or operate independently to generate two separate 32-bit permutation results.

4) Shift Unit (SU): Some block ciphers use shift operations to increase the complexity of deducing key values from encrypted data. For example, in AES, each row of a 4×4 data block is shifted by a different offset to shuffle data locations [13]. The SU supports such operations and consists of two barrel shifters. These shifters can be combined to support up to 64-bit arithmetic, logical, and rotate shift operations.

5) Table Unit (TU): To obscure the statistical properties of the plaintext, the S-Box maps input bits to corresponding output bits using a predefined substitution table. We implement the S-Box operation using lookup tables. Specifically, the TU includes four S-Box units, each with a 256-entry byte-wide lookup table. These tables primarily convert 8-bit input data into 8-bit output values. For wider input data, multiple tables can be combined to function as a single S-Box. For narrower input data, a single table can perform substitution with zero-padding on the most significant bits.

B. Asymmetric Cryptographic Engine (ACE)

Fig. 11-(a) depicts the structure of ACE, designed for digital signature generation and verification. ACE integrates a Hash ALU cluster for hash computations and two asymmetric cipher ALUs for public-key and post-quantum cryptography.

1) Hash ALU Cluster: Digital signature algorithms typically hash messages, then sign the resulting digest [22]. Hashing enables efficient handling of arbitrary-length messages by reducing them to fixed-size digests. Furthermore, certain PQC schemes like SPHINCS+ [8] rely on hash functions as core primitives. To support conventional signature hashing and hash-based PQC, ACE integrates a dedicated Hash ALU cluster. Hash functions are inherently parallelizable [6] as they process input by dividing it into fixed-size blocks and operating independently via tree hashing [6]. To support this parallelism, FlashVault incorporates eight Hash ALUs to accelerate the hashing process.

Fig. 11-(b) shows the block diagram of the Hash ALU, which comprises four key components optimized for the primitive operations used in cryptographic hash functions: (i) a modular addition unit, (ii) a logic operation unit, (iii) a permutation unit, and (iv) a shift unit. The *modular addition unit* supports both standard and modular additions for hash state updates. The *logic operation unit* supports bitwise operations such as XOR, AND, and NOT to introduce non-linearity by disrupting linear relationships between input and output. The *permutation unit*, based on a Benes network, reorders bit positions to enhance diffusion. The *shift unit* supports word-level (e.g., 32- or 64-bit) logical and circular shifts, as commonly used in hash functions like SHA-2 [76].

2) Asymmetric Cipher ALUs: PKC and PQC computations typically proceed sequentially, with each stage depending on prior results, e.g., modular exponentiation, scalar multiplication, and NTT/FFT-based transforms. To efficiently support this, ACE integrates two asymmetric cipher ALUs, which include the arithmetic, logic, permutation, and shift units. These units handle high-precision integer arithmetic, bit manipulations, and data transformations. While architecturally similar to the Hash ALU, asymmetric cipher ALUs support more diverse and complex arithmetic for PKC/PQC. That is, the *arithmetic unit* includes extended functional blocks beyond the modular addition unit of the Hash ALU, including adders, multipliers, and modular operators. The arithmetic unit adopts a fixed 64-bit integer datapath, zero-extending narrower operands for alignment. For high-precision computations, operands are divided into 64-bit limbs (subwords), with limb-wise computation involving carry propagation and cross-limb alignment. The adder in the arithmetic unit is designed with a carry-save structure to efficiently support multi-limb accumulation. To reduce propagation delay, the multiplier uses a Wallace tree [120] and the modular unit uses Barrett reduction [54] with shifters and fixed-coefficient multipliers.

VII. Algorithm Selection and Reconfiguration

To select a cryptographic algorithm in FlashVault, a user-level application issues an `ioctl()` system call with the desired algorithm identifier as an argument. This transfers an NVMe command containing the feature data of the selected algorithm to the system buffer in the kernel space. The device driver then issues an `nvme_set_feature` command, which updates control registers in the SSD controller. These registers propagate a control signal to the FSM unit in FlashVault, which interprets the selected algorithm and reconfigures the cryptographic pipeline by enabling the necessary modules and associated datapaths. This hardware-level reconfiguration is performed dynamically at runtime, enabling FlashVault to support diverse cryptographic algorithms without requiring firmware updates or system reboot.

VIII. Evaluation

A. Area and Power Analysis of FlashVault

1) Methodology: To evaluate the effectiveness of FlashVault, we synthesized all components shown in Fig. 9 and performed place-and-route (P&R) using Synopsys EDA tools at fast/fast and slow/slow corners in a 65nm CMOS process. For low-power design, we utilized the unified power format (UPF) to implement power gating. Additionally, for dynamic power minimization, we implemented clock gating at the RTL level.

TABLE II

AREA AND POWER OF THE HARDWARE COMPONENTS IN FLASHVAULT

Block Cipher Engine (Bottom Level)			
Component	**Configuration**	**Area [μm^2]**	**Power [mW]**
Arithmetic Unit	2 × Arithmetic Logic	1,263.15	0.10
Logical Operation Unit	2 × Logic Cell Block	99.93	0.02
Permutation Unit	2 × Benes Network	247.19	0.08
Shift Unit	2 × Barrel Shifter	205.83	0.05
Table Unit	2 × S-Box Unit	2,195.22	1.17
Total		**4,2616.21**	**0.41**
Asymmetric Cipher Engine (Bottom Level)			
Component	**Configuration**	**Area [μm^2]**	**Power [mW]**
Hash ALU Cluster	8 × Hash ALU,	38,354.59	3.43
	1 Padding Unit	1,491.57	0.06
Asymmetric Cipher ALU Pair (2 ACALUs)	2 × ACALU,	13,540.39	1.65
	1 Padding Unit	1,605.35	0.08
Total		**54,991.89**	**5.13**
FlashVault (Top Level)			
Component	**Configuration**	**Area [μm^2]**	**Power [mW]**
Control Unit	1 FSM	399.24	0.09
Buffer Unit	1 Page Bufffer (4 KB), 1 Cache Buffer (4 KB)	9,024.34	1.12
Interface Unit	1 FIFO, 1 Router	2,538.33	2.31
Cryptographic Engine	1 LDPC Decoder	58,624.29	6.65
	1 Key Generation Unit	2,321.01	0.21
	2 × Registers (1 KB)	2,654.89	0.19
	1 Output Register (512 B)	1624.79	0.13
	1 Block Cipher Engine Array	64,180.91	6.63
	1 Asymmetric Cipher Engine	54,991.85	5.13
Total		**54,991.89**	**20.71**

On-chip memories were instantiated using commercial memory IPs. Area was extracted using Synopsys IC Compiler II [109] and consumed power was estimated using PrimeTime PX [110] based on switching activity interchange format (SAIF) generated from post-layout simulations with StarRC-extracted parasitics [111]. To report results under more advanced technology assumptions, we scaled the area and power consumption to 14nm technology node using scaling factors from [40]. This node selection reflects the technology used in recent commercial SSDs, which commonly adopt nodes in the mid-10nm range, i.e., 14nm [25], [31], [61].

2) Analysis with Synthesized Results: Table II summarizes the post-layout area and power of each submodule in FlashVault. The control unit, buffer unit, and interface unit occupy 399.24, 9,024.34, and 2,538.33 μm^2, respectively, and consume 0.09, 1.12, and 2.31 mW of power. In addition, the cryptographic engine occupies 0.18 mm^2 and consumes 18.87 mW. Placing cryptographic engines in parallel at the die level enhances I/O throughput. Accordingly, we estimate the maximum number of engines (N) that can be integrated within a 4D V-NAND chip as follow:

$$N = \min\left(\left\lfloor \frac{A_{\text{budget}}}{A_{\text{engine}}} \right\rfloor, \left\lfloor \frac{P_{\text{budget}}}{P_{\text{engine}}} \right\rfloor\right), \tag{2}$$

where A_{budget} and P_{budget} denote the area and power budgets, while A_{engine} and P_{engine} represent the area and power of a single cryptographic engine. By using post-layout results and budgets from Section IV-A and IV-B, we find that up to two cryptographic engines can be integrated in parallel. This translates to 32 BCEs, 16 Hash ALUs, and 4 asymmetric cipher ALUs that can be integrated within the unused area of a 4D V-NAND chip.

B. Experimental Setup for Architectural Evaluation

1) FlashVault Implementation: As described in Section VIII-A, FlashVault integrates two cryptographic engines within the area and power budgets of Samsung's 7th-generation V-NAND. Based on this configuration, we evaluate the in-die cryptographic performance. FlashVault is implemented using SimpleSSD [32], incorporating the P&R results in Table II and the system parameters in Table III.

TABLE III

SIMULATED SSD CONFIGURATIONS

Component	**Configuration**
Controller	Arm Cortex-R8 Processor (@ 1.5 GHz) [5] 16 Gb LPDDR3 [69]
Flash Memory	4 Channels, 4 Packages, 2 Dies, 4 Planes 682 Blocks, 128 Pages, Page Size = 4 KB NV-DDR3, 1600 MT/s, 8-bit Channel Bus
Cryptographic Unit	2 Cryptographic Engines (@ 200 MHz) (32 BCEs, 16 Hash ALUs, and 4 Asymmetric Cipher Engines)
Timing Parmeters	t_R = 45 μs, t_{PROG} = 400 μs, t_{ERASE} = 2 ms t_{PCBSY}, t_{RCBSY} = 3 μs, Storage Stack Latency = 5 μs

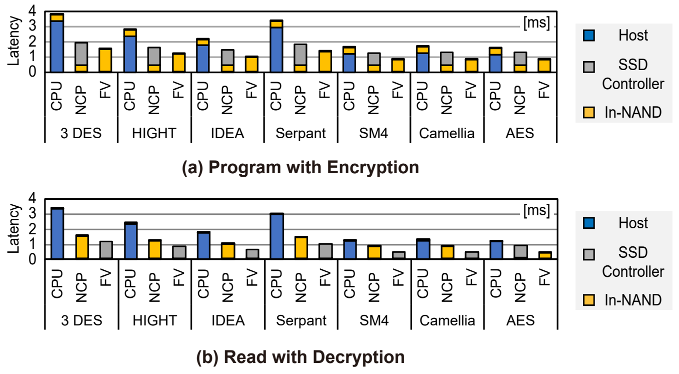

Fig. 12. (a) Program and (b) read latency breakdowns for various block ciphers using CPU, NCP, and FlashVault (FV) with 256 KB input.

2) Baseline Systems: We compare FlashVault with two baselines: (i) a Near-Core Processing (NCP) architecture and (ii) a real-machine system (CPU + SSD). The *NCP architecture* uses the same cryptographic engines as FlashVault with identical configuration, but places them near the SSD controller instead of inside the NAND die. Following recent NCP designs [49], [62], [66], [85], our NCP baseline uses a device DRAM as working memory for encryption. The real-machine system performs encryption on the host CPU which follows the hardware setup provided in Section III-B1.

3) Methodology: We modified the hardware abstraction layer of SimpleSSD [32] to enable end-to-end latency estimation for NVMe requests. To reflect the latency overhead of cryptographic operations in FlashVault and NCP, we injected cycle-level latency estimated from design-for-test (DFT) verification using Synopsys tools into the I/O request handling path and controller-side I/O processing path of SimpleSSD, respectively. In addition, we modified the DRAM abstraction layer in the NCP model to reflect the DRAM access latency during cryptographic operations. The real-machine system follows the methodology described in Section III-B1, with SSD latency derived from baseline SimpleSSD (without cryptographic engines) for fair comparison. Our evaluation uses the block ciphers, PKC, and PQC algorithms listed in Section III-B as benchmarks, assessed by varying the input sizes.

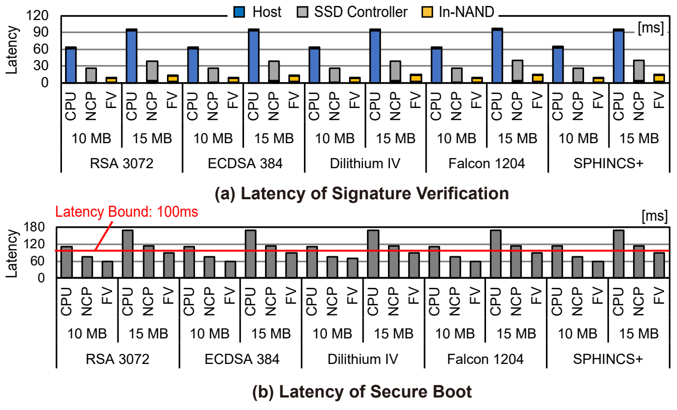

Fig. 13. Latencies of (a) signature verification and (b) end-to-end secure boot, evaluated with 10 MB and 15 MB firmware images.

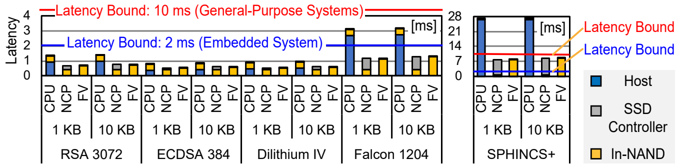

Fig. 14. Latency of signature generation evaluated with 1 KB and 10 KB inputs. Red and blue lines indicate latency bounds for anti-tamper logging in general-purpose and embedded systems, respectively.

C. Performance Evaluation

1) Block Cipher: Fig. 12 presents the latency of program and read operations for 256 KB input data. Among the three systems, encryption on the host CPU incurs the highest latency due to data transfer and software overhead. The NCP architecture employs dedicated hardware near the SSD controller and removes software stack overhead, resulting in average latency improvements of 36.9% and 43.2% for program and read operations, respectively, over the real-machine system. FlashVault integrates cryptographic engines within the NAND die and performs encryption directly without traversing the DRAM, achieving additional latency reductions of 28.8% and 37.5%, on average, for program (with encryption) and read (with decryption) operations, respectively, compared to NCP.

2) Signature Verification: Fig. 13-(a) presents the signature verification latency for 10 MB and 15 MB firmware images. FlashVault achieves 1.9× and 1.3× lower latency, on average, compared to the real-machine system and the NCP architecture, respectively. Fig. 13-(b) shows the total latency of secure boot, including boot code loading. As discussed in Section III-A, commercial SSDs are typically required to complete secure boot within 100 ms after power stabilization. The host CPU–based verification fails to meet this constraint for all algorithms across the tested firmware sizes. The NCP architecture meets the constraint for all algorithms at 10 MB but fails at 15 MB. In contrast, FlashVault completes secure boot within 100 ms for all algorithms and image sizes, demonstrating its capability to securely boot even large firmware images used to support advanced features.

3) Signature Generation: Fig. 14 shows the signature generation latency for 1 KB and 10 KB inputs. FlashVault achieves 2.9× and 3.1× lower latency on average than the real-machine system for 1 KB and 10 KB, respectively. Anti-tamper logging, a representative application of signature generation, is typically required to complete within 2 to 15 ms in commercial SSDs, as discussed in Section III-B4. Specifically, real-time embedded systems typically require completion within 2 ms [99], while general-purpose systems tolerate up to 15 ms [1]. However, the real-machine system exceeds the latency bounds of both systems when executing PQC algorithms. In contrast, FlashVault meets the general-purpose latency bounds for all PQC algorithms and satisfies the stricter embedded constraint for all PQC algorithms except SPHINCS+. This indicates that FlashVault supports most cryptographic algorithms in both embedded and general-purpose environments. Since the evaluation uses small input sizes representative of typical log entries, the DRAM access overhead in the NCP is negligible, resulting in a modest performance gain of 0.2 to 1.0% for FlashVault over NCP.

4) FTL Overhead: All prior evaluations were conducted under best-case conditions, with the system freshly initialized and no background FTL activities such as GC and WL. In this state, the baseline FTL latency is approximately 4.6 μs for a 1 KB input and increases sub-linearly with larger input sizes. This trend is attributed to the proportional growth in processing time for address translation and data buffering. To assess the impact of GC and WL, we implemented a steady-state environment in SimpleSSD [32] by filling the SSD with dummy data and repeatedly overwriting random locations. Table IV presents latency and best-case-relative overhead under steady-state conditions, using the same inputs and workloads as in the previous evaluation. For block ciphers, program operations incur an average FTL overhead of 46.1%, while read operations show a higher 79.3% overhead due to the relatively short NAND read latency, which makes the FTL more dominant in total latency. For public-key operations, signature verification and signing show average overheads of 3.6% and 127.6%, respectively. The verification incurs minimal overhead, as cryptographic latency dominates, whereas signing suffers from high overhead due to shorter execution time and smaller inputs. SPHINCS+ deviates from the trend in signing overheads due to substantial cryptographic latency, which is caused by hash tree traversal and high computational complexity. Although not shown in the table, verification with a 10 MB input and signing with a 1 KB input result in average

TABLE IV
LATENCY OF CRYPTOGRAPHIC OPERATIONS IN A STEADY-STATE SSD

Scheme	Operation	Input Size	Cryptographic Algorithms (Overall Latency [ms]; Overhead [%])
Block Cipher	Program (w/ Encrypt)	256 KB	3 DES (2.0; 31.7), HIGHT (1.7; 39.4), IDEA (1.5; 46.6), Serpant (1.9; 35.3), SM4 (1.3; 55.9), Camellia (1.3; 55.6), AES (1.3; 58.0)
	Read (w/ Decrypt)	256 KB	3 DES (1.6; 38.9), HIGHT (1.3; 53.2), IDEA (1.1; 68.5), Serpant (1.5; 43.8), SM4 (0.9; 92.9), Camellia (0.9; 90.1), AES (0.9; 96.2)
PKC & PQC	Signature Verification	15 MB	RSA 3072(13.5; 3.5), ECDSA 384 (13.5; 3.6), Dilitium IV (13.6, 3.5), Falcon 1024 (13.8; 3.5) , SPHINCS + (13.7, 3.5)
	Signature Generation	10 KB	RSA 3072(1.6; 116.7), ECDSA 384 (1.5; 154.0), Dilitium IV (1.5; 155.3), Falcon 1024 (2.2; 76.1) , SPHINCS + (9.5; 12.1)

GC/WL overheads of 5.6% and 87.6%, respectively.

IX. Conclusion

We present FlashVault, a novel in-NAND self-encryption architecture that embeds reconfigurable cryptographic engines directly beneath the 4D NAND flash, enabling secure data processing with zero area overhead. By leveraging unused silicon area under vertically-stacked flash cell arrays, FlashVault supports diverse cryptographic algorithms, including block ciphers, public-key cryptography, and post-quantum cryptography, without relying on the host CPU. To enable this reconfigurability, we thoroughly broke down compute primitives in various cryptographic algorithms, so that the cryptographic engine in FlashVault can support all required primitives with reusable compute units. FlashVault achieves $1.46\sim3.45\times$ and $1.02\sim2.01\times$ performance gains over host-based encryption and near-core processing architectures (i.e., with cryptographic engines placed near the SSD controller), respectively, across diverse cryptographic algorithms.

References

[1] Ahmad, Adil and Lee, Sangho and Peinado, Marcus, "HARDLOG: Practical Tamper-Proof System Auditing Using a Novel Audit Device," in *IEEE Symposium on Security and Privacy (SP)*, 2022, pp. 1791–1807.

[2] S. Ahmadian, F. Taheri, M. Lotfi, M. Karimi, and H. Asadi, "Investigating power outage effects on reliability of solid-state drives," in *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2018, pp. 207–212.

[3] N. Y. Ahn and D. H. Lee, "Adaptive privacy-preserving SSD," 2025. [Online]. Available: https://arxiv.org/abs/2506.02030

[4] Aoki, Kazumaro and Ichikawa, Tetsuya and Kanda, Masayuki and Matsui, Mitsuru and Moriai, Shiho and Nakajima, Junko and Tokita, Toshio, "Camellia: A 128-Bit Block Cipher Suitable for Multiple Platforms - Design and Analysis," in *International Workshop on Selected Areas in Cryptography (SAC)*, 2000, p. 39–56.

[5] ARM, "Cortex-R8," https://developer.arm.com/Processors/Cortex-R8, [Online; accessed 08-July-2025].

[6] K. Atighehchi and R. Rolland, "Optimization of tree modes for parallel hash functions: A case study," *IEEE Transactions on Computers (TC)*, vol. 66, no. 9, pp. 1585–1598, 2017.

[7] S. Bai, L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, P. Schwabe, G. Seiler, and D. Stehlé, "CRYSTALS-Dilithium Algorithm Specifications and Supporting Documentation," https://pq-crystals.org/dilithium/data/dilithium-specification-round3-20210208.pdf, 2021, [Online; accessed 02-July-2025].

[8] D. J. Bernstein, A. Hülsing, S. Kölbl, R. Niederhagen, J. Rijneveld, and P. Schwabe, "The SPHINCS+ Signature Framework," in *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2019, p. 2129–2146.

[9] Biham, Eli and Anderson, Ross J. and Knudsen, Lars R., "Serpent: A New Block Cipher Proposal," in *International Workshop on Fast Software Encryption (FSE)*, 1998, p. 222–238.

[10] Billy Tallis, "2021 nand flash updates from isscc: The learning towers of tlc and qlc," https://www.anandtech.com/show/16491/flash-memory-at-isscc-2021, 2021, [Online; accessed 19-June-2025].

[11] Botan, "Botan: Crypto and TLS for Modern C++," https://botan.randombit.net/, 2024, [Online; accessed 02-June-2025].

[12] Y. Cai, S. Ghose, E. F. Haratsch, Y. Luo, and O. Mutlu, "Error Characterization, Mitigation, and Recovery in Flash-Memory-Based Solid-State Drives," *Proceedings of the IEEE*, vol. 105, pp. 1666–1704, 2017.

[13] X. Chen, "Implementing AES Encryption on Programmable Switches via Scrambled Lookup Tables," in *ACM SIGCOMM Workshop on Secure Programmable Network Infrastructure (SPIN)*, 2020, p. 8–14.

[14] J. Cho, D. C. Kang, J. Park, S.-W. Nam, J.-H. Song, B.-K. Jung, J. Lyu, H. Lee, W.-T. Kim, H. Jeon, S. Kim, I.-M. Kim, J.-I. Son, K. Kang, S.-W. Shim, J. Park, E. Lee, K.-M. Kang, S.-W. Park, J. Lee, S. H. Moon, P. Kwak, B. Jeong, C. A. Lee, K. Kim, J. Ko, T.-H. Kwon, J. Lee, Y. Lee, C. Kim, M.-W. Lee, J.-y. Yun, H. Lee, Y. Choi, S. Hong, J. Park, Y. Shin, H. Kim, H. Kim, C. Yoon, D. S. Byeon, S. Lee, J.-Y. Lee, and J. Song, "30.3 A 512Gb 3b/Cell 7th -Generation 3D-NAND Flash Memory with 184MB/s Write Throughput and 2.0Gb/s Interface," in *IEEE International Solid-State Circuits Conference (ISSCC)*, vol. 64, 2021, pp. 426–428.

[15] E.-S. Choi and S.-K. Park, "Device considerations for high density and highly reliable 3D NAND flash cell in near future," in *International Electron Devices Meeting (IEDM)*, 2012, pp. 9.4.1–9.4.4.

[16] M. H. I. Chowdhuryy, M. Jung, F. Yao, and A. Awad, "D-shield: Enabling processor-side encryption and integrity verification for secure nvme drives," in *IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, 2023, pp. 908–921.

[17] M. Chun, J. Lee, M. Kim, J. Park, and J. Kim, "Rif: Improving read performance of modern ssds using an on-die early-retry engine," in *IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, 2024, pp. 643–656.

[18] M. Chun, J. Lee, S. Lee, M. Kim, and J. Kim, "Pif: in-flash acceleration for data-intensive applications," in *ACM Workshop on Hot Topics in Storage and File Systems (HotStorage)*, 2022, p. 106–112.

[19] F. T. Commission, "Equifax data breach settlement," https://consumer.ftc.gov/consumer-alerts/2019/07/equifax-data-breach-settlement-what-you-should-know, 2019, online; accessed 17-June-2025.

[20] M. . Company, "The value of getting personalization right—or wrong—is multiplying," https://www.mckinsey.com/business-functions/marketing-and-sales/our-insights/the-value-of-getting-personalization-right-or-wrong-is-multiplying, 2021, online; accessed 17-June-2025.

[21] Cui, Hangxuan and Lin, Jun and Wang, Zhongfeng, "An Improved Gradient Descent Bit-Flipping Decoder for LDPC Codes," *IEEE Transactions on Circuits and Systems I: Regular Papers (TCAS-I)*, vol. 66, no. 8, pp. 3188–3200, 2019.

[22] Q. Dang, "Recommendation for applications using approved hash algorithms," https://doi.org/10.6028/NIST.SP.800-107r1, 2009, [Online; accessed 16-July-2025].

[23] Daniel Hounslow, "Japan-Data Protection Overview," https://www.dataguidance.com/notes/japan-data-protection-overview, 2023, [Online; accessed 24-June-2025].

[24] DataGuidance, "Japan - Data Protection Overview," https://www.dataguidance.com/notes/japan-data-protection-overview, 2023, [Online; accessed 20-June-2025].

[25] Dick James and Jeongdong Choe, "NAND Flash Technology," https://www.techinsights.com/ko/node/30189, 2019, [Online; accessed 08-June-2025].

[26] L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, P. Schwabe, G. Seiler, and D. Stehlé, "CRYSTALS-Dilithium: A Lattice-Based Digital Signature Scheme," in *IACR Transactions on Cryptographic Hardware and Embedded Systems (TCHES)*, 2018, pp. 238–268.

[27] D. EMC, "Self-encrypting drives in dell emc poweredge servers with vmware vsphere," https://dl.dell.com/manuals/common/vmware_esxi_6x_7x_whitepaper7_en_us.pdf, 2020, accessed: July 7, 2025.

[28] European Parliament, "General Data Protection Regulation (GDPR)," https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32016R0679, 2016, [Online; accessed 20-June-2025].

[29] Federal Office for Information Security (BSI), "Elliptic Curve Cryptography," https://www.techpowerup.com/ssd-specs/sk-hynix-gold-p31-2-tb.d443, 2018, [Online; accessed 25-June-2025].

[30] J. L. Galea, E. D. Mulder, D. Page, and M. Tunstall, "Soc it to em: Electromagnetic side-channel attacks on a complex system-on-chip," in *Cryptographic Hardware and Embedded Systems (CHES)*, ser. Lecture Notes in Computer Science, vol. 9293, 2015, pp. 620–640.

[31] A. Goda, "Recent Progress on 3D NAND Flash Technologies," *Electronics*, vol. 10, no. 24, 2021.

[32] D. Gouk, M. Kwon, J. Zhang, S. Koh, W. Choi, N. S. Kim, M. Kandemir, and M. Jung, "Amber: Enabling Precise Full-System Simulation with Detailed Modeling of All SSD Resources," in *IEEE/ACM International Symposium on Microarchitecture (MICRO)*, 2018, pp. 469–481.

[33] Haoran Gong and Tailiang Ju, "Distributed power analysis attack on SM4 encryption chip," *Scientific Reports* , vol. 14, no. 1007, 2024.

[34] Hartmut Lonzer, Corne Lottering, and Vasfi Gucer, "IBM Storage FlashSystem 5200 Product Guide for IBM Storage Virtualize 8.6," https://www.redbooks.ibm.com/redpapers/pdfs/redp5617.pdf, 2023, [Online; accessed 20-June-2025].

[35] HashedOut, "Cost of 2013 target data breach nears $3 million," https://www.thesslstore.com/blog/2013-target-data-breach-settled/, year = 2017, note = Online; accessed 17-June-2025,.

[36] G. Hatzivasilis, K. Fysarakis, I. Papaefstathiou, and C. Manifavas, "A review of lightweight block ciphers," *Journal of Cryptographic Engineering (J. Cryptogr. Eng.)*, vol. 8, no. 2, pp. 141–184, 2018.

[37] D. Hong, J. Sung, S. Hong, J. Lim, S. Lee, B.-S. Koo, C. Lee, D. Chang, J. Lee, K. Jeong, H. Kim, J. Kim, and S. Chee, "HIGHT: a new block cipher suitable for low-resource device," in *International Conference on Cryptographic Hardware and Embedded Systems (CHES)*, 2006, p. 46–59.

[38] C.-L. Hsiung, Y. An, A. Chu, and F. Toyama, "Architecture for CMOS under array," in *US9922716B2 (US Patent)*, 2018.

[39] F.-C. Hsu and S. Jose, "3D cell and array structures," U.S. Patent Application US2024/0404598 A1, 2024.

[40] W. Huang, K. Rajamani, M. R. Stan, and K. Skadron, "Scaling with Design Constraints: Predicting the Future of Big Chips," *IEEE Micro*, vol. 31, no. 4, pp. 16–29, 2011.

[41] IEEE, "IEEE Standard for Discovery, Authentication, and Authorization in Host Attachments of Storage Devices," *IEEE Std 1667-2015 (Revision of IEEE Std 1667-2009)*, pp. 1–230, 2016.

[42] IEEE, "IEEE Standard for Cryptographic Protection of Data on Block-Oriented Storage Devices," *IEEE Std 1619-2018 (Revision of IEEE Std 1619-2007)*, pp. 1–41, 2019.

[43] T. Insights, "Comparison: Latest 3D NAND Products from YMTC, Samsung, SK hynix and Micron," https://www.techinsights.com/ko/node/49939, 2023, [Online; accessed 23-June-2025].

[44] Intel, "What Type of Encryption Do Intel® SSDs Use?" https://www.intel.com/content/www/us/en/support/articles/000036098/memory-and-storage.html, 2022, [Online; accessed 20-June-2025].

[45] K. Nasim, E. Pang, J. Carreno, S. Kareti, K. C., R. Wahler, E. Marando, R. Rao, R. Kumar, and R. Schoepflin, "¡Project Ersa AMC¿ Design Specification," https://www.opencompute.org/documents/project-ersa-amc-ocp-draft-v1-0-final-pdf-1, 2024, [Online; accessed 01-July-2025].

[46] D. Kang, M. Kim, S. C. Jeon, W. Jung, J. Park, G. Choo, D.-k. Shim, A. Kavala, S.-B. Kim, K.-M. Kang, J. Lee, K. Ko, H.-W. Park, B.-J. Min, C. Yu, S. Yun, N. Kim, Y. Jung, S. Seo, S. Kim, M. K. Lee, J.-Y. Park, J. C. Kim, Y. S. Cha, K. Kim, Y. Jo, H. Kim, Y. Choi, J. Byun, J.-h. Park, K. Kim, T.-H. Kwon, Y. Min, C. Yoon, Y. Kim, D.-H. Kwak, E. Lee, W.-g. Hahn, K.-s. Kim, K. Kim, E. Yoon, W.-T. Kim, I. Lee, S. h. Moon, J. Ihm, D. S. Byeon, K.-W. Song, S. Hwang, and K. H. Kyung, "13.4 A 512Gb 3-bit/Cell 3D 6th-Generation V-NAND Flash Memory with 82MB/s Write Throughput and 1.2Gb/s Interface," in *IEEE International Solid-State Circuits Conference (ISSCC)*, 2019, pp. 216–218.

[47] L. Kang, Y. Xue, W. Jia, X. Wang, J. Kim, C. Youn, M. J. Kang, H. J. Lim, B. Jacob, and J. Huang, "Iceclave: A trusted execution environment for in-storage computing," in *IEEE/ACM International Symposium on Microarchitecture (MICRO)*, 2021, pp. 199–211.

[48] J. Kim, M. Jung, and J. Kim, "Decoupled ssd: Rethinking ssd architecture through network-based flash controllers," in *ACM/IEEE International Symposium on Computer Architecture (ISCA)*, 2023.

[49] J. Kim, M. Kang, Y. Han, Y.-G. Kim, and L.-S. Kim, "OptimStore: In-Storage Optimization of Large Scale DNNs with On-Die Processing," in *IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, 2023, pp. 611–623.

[50] M. Kim, J. Park, G. Cho, Y. Kim, L. Orosa, O. Mutlu, and J. Kim, "Evanesco: Architectural Support for Efficient Data Sanitization in Modern Flash-Based Storage Systems," in *ACM Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, 2020, p. 1311–1326.

[51] S. S. Kim, S. K. Yong, W. Kim, S. Kang, H. W. Park, K. J. Yoon, D. S. Sheen, S. Lee, and C. S. Hwang, "Review of Semiconductor Flash Memory Devices for Material and Process Issue," *Advanced Materials*, vol. 35, no. 2200659, 2022.

[52] Kingston Technology, "What is SSD Encryption and How Does It Work?" https://www.kingston.com/en/blog/data-security/how-ssd-encryption-works, 2021, [Online; accessed 20-June-2025].

[53] Kişisel Verileri Koruma Kanunu (KVKK), "Personal Data Protection Authority," https://www.kvkk.gov.tr/SharedFolderServer/CMSFiles/1d7c2f99-be2c-4971-a304-0a1eb3586bd1.pdf, 2024, [Online; accessed 24-June-2025].

[54] M. Knezevic, F. Vercauteren, and I. Verbauwhede, "Faster interleaved modular multiplication based on barrett and montgomery reduction methods," *IEEE Transactions on Computers (TC)*, vol. 59, no. 12, pp. 1715–1721, 2010.

[55] N. Koblitz and A. J. Menezes, "A survey of public-key cryptosystems," *SIAM Review*, vol. 46, no. 4, p. 599–634, 2004.

[56] Korea Internet and Security Agency (KISA), "Korean Cryptographic Module Validation Program," https://seed.kisa.or.kr/kisa/kcmvp/EgovVerification.do, 2023, [Online; accessed 24-June-2025].

[57] Korea Legislation Research Institute, "Personal Information Protection Act," https://elaw.klri.re.kr/eng_service/lawView.do?hseq=62389&lang=ENG, 2023, [Online; accessed 20-June-2025].

[58] A. Kumar, R. Tandon, and T. C. Clancy, "On the latency and energy efficiency of distributed storage systems," *IEEE Transactions on Cloud Computing (TCC)*, vol. 5, no. 2, pp. 221–233, 2017.

[59] Lai, Xuejia and Massey, James L., "A proposal for a new block encryption standard," in *Workshop on the theory and application of cryptographic techniques on Advances in cryptology (EUROCRYPT)*, 1991, p. 389–404.

[60] J.-D. Lee, S.-H. Hur, and J.-D. Choi, "Effects of floating-gate interference on NAND flash memory cell operation," *IEEE Electron Device Letters (EDL)*, vol. 23, pp. 264–266, 2002.

[61] S. Lee, J.-y. Lee, I.-h. Park, J. Park, S.-w. Yun, M.-s. Kim, J.-h. Lee, M. Kim, K. Lee, T. Kim, B. Cho, D. Cho, S. Yun, J.-n. Im, H. Yim, K.-h. Kang, S. Jeon, S. Jo, Y.-l. Ahn, S.-M. Joe, S. Kim, D.-k. Woo, J. Park, H.-w. Park, Y. Kim, J. Park, Y. Choi, M. Hirano, J.-D. Ihm, B. Jeong, S.-K. Lee, M. Kim, H. Lee, S. Seo, H. Jeon, C.-h. Kim, H. Kim, J. Kim, Y. Yim, H. Kim, D.-S. Byeon, H.-J. Yang, K.-T. Park, K.-h. Kyung, and J.-H. Choi, "A 128Gb 2b/cell NAND flash memory in 14nm technology with tPROG=640μs and 800MB/s I/O rate," in *IEEE International Solid-State Circuits Conference (ISSCC)*, 2016, pp. 138–139.

[62] S. Li, F. Tu, L. Liu, J. Lin, Z. Wang, Y. Kang, Y. Ding, and Y. Xie, "ECSSD: Hardware/Data Layout Co-Designed In-Storage-Computing Architecture for Extreme Classification," in *ACM/IEEE Annual International Symposium on Computer Architecture (ISCA)*, 2023.

[63] Ling Song and Guohong Liao and Jian Guo, "Non-full Sbox Linearization: Applications to Collision Attacks on Round-Reduced Keccak," in *Advances in Cryptology (CRYPTO)*, 2017, pp. 428–451.

[64] Y. Luo, S. Ghose, Y. Cai, E. F. Haratsch, and O. Mutlu, "HeatWatch: Improving 3D NAND Flash Memory Device Reliability by Exploiting Self-Recovery and Temperature Awareness," in *IEEE International Symposium on High Performance Computer Architecture (HPCA)*, 2018, pp. 504–517.

[65] Y. Luo, S. Ghose, Y. Cai, E. F. Haratsch, and O. Mutlu, "Improving 3D NAND Flash Memory Lifetime by Tolerating Early Retention Loss and Process Variation," *ACM on Measurement and Analysis of Computing Systems (POMACS)*, vol. 2, no. 3, 2018.

[66] R. Mahapatra, H. Santhanam, C. Priebe, H. Xu, and H. Esmaeilzadeh, "In-storage acceleration of retrieval augmented generation as a service," in *ACM/IEEE International Symposium on Computer Architecture (ISCA)*, 2025, pp. 450–466.

[67] A. Maiti and P. Schaumont, "Improved ring oscillator puf: An fpga-friendly secure primitive," *Journal of Cryptology*, vol. 24, no. 2, p. 375–397, 2011.

[68] C. Meijer and B. van Gastel, "Self-Encrypting Deception: Weaknesses in the Encryption of Solid State Drives," in *IEEE Symposium on Security and Privacy (SP)*, 2019, pp. 72–87.

[69] Micron, "Mobile LPDDR3 SDRAM: 178-Ball, Single-Channel Mobile LPDDR3 SDRAM Features," www.micron.com/products/dram/lpdram/16Gb, 2014.

[70] Micron, "MICRON 1100 SSD FIPS 140-2 Cryptographic Module Non-Proprietary Security Policy," https://csrc.nist.gov/csrc/media/projects/cryptographic-module-validation-program/documents/security-policies/140sp2848.pdf, 2024, [Online; accessed 20-June-2025].

[71] C. C. Moallemi and İ. Sağlam, "Or forum—the cost of latency in high-frequency trading," *Operations Research*, vol. 61, no. 5, pp. 1070–1086, 2013.

[72] V. Mohan, S. Gurumurthi, and M. R. Stan, "Flashpower: A detailed power model for nand flash memory," in *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2010, pp. 502–507.

[73] National Institute of Standards and Technology (NIST), "Security Requirements for Cryptographic Modules," https://csrc.nist.gov/pubs/fips/140-2/upd2/final, 2002, [Online; accessed 24-June-2025].

[74] National Institute of Standards and Technology (NIST), "Security Requirements for Cryptographic Modules," https://csrc.nist.gov/pubs/fips/140-2/upd2/final, 2019, [Online; accessed 24-June-2025].

[75] National Institute of Standards and Technology (NIST), "NIST Releases First 3 Finalized Post-Quantum Encryption Standards," https://www.nist.gov/news-events/news/2024/08/nist-releases-first-3-finalized-post-quantum-encryption-standards, 2024, [Online; accessed 20-June-2025].

[76] National Institute of Standards and Technology (NIST), "Secure Hash Standard (SHS)," https://files.floridados.gov/media/704729/fips-pub-180.pdf, 2024, [Online; accessed 06-June-2025].

[77] National Institute of Standards and Technology (NIST), "Cryptographic Module Validation Program (CMVP)," https://csrc.nist.gov/projects/cryptographic-module-validation-program, 2025, [Online; accessed 24-June-2025].

[78] H. Nejatollahi, N. Dutt, S. Ray, F. Regazzoni, I. Banerjee, and R. Cammarota, "Post-Quantum Lattice-Based Cryptography Implementations: A Survey," *ACM Computing Surveys (CSUR)*, vol. 51, no. 6, pp. 1–41, 2019.

[79] OECD, "The path to becoming a data-driven public sector," https://doi.org/10.1787/059814a7-en, 2019, online; accessed 17-June-2025.

[80] D. ohnson, A. Menezes, and S. Vanstone, "The Elliptic Curve Digital Signature Algorithm (ECDSA)," *International Journal of Information Security (Int. J. Inf. Secur.)*, vol. 1, no. 1, pp. 36–63, 2001.

[81] Open Quantum Safe, "liboqs," https://openquantumsafe.org/liboqs/, 2024, [Online; accessed 02-July-2025].

[82] OpenSSL, "The Open Source toolkit for SSL/TLS," https://www.openssl.org/, 1998, [Online; accessed 02-July-2025].

[83] Osvik, Dag Arne and Shamir, Adi and Tromer, Eran, "Cache Attacks and Countermeasures: The Case of AES," in *IEEE Symposium on Security and Privacy (SP)*, 2006, pp. 200–213.

[84] P. Karn and B. Kaliski, "The ESP Triple DES Transform," https://www.rfc-editor.org/rfc/rfc1851.html, 1995, [Online; accessed 02-July-2025].

[85] X. Pan, Y. An, S. Liang, B. Mao, M. Zhang, Q. Li, M. Jung, and J. Zhang, "Flagger: Cooperative acceleration for large-scale cross-silo federated learning aggregation," in *ACM/IEEE International Symposium on Computer Architecture (ISCA)*, 2024, pp. 915–930.

[86] J.-W. Park, D. Kim, S. Ok, J. Park, T. Kwon, H. Lee, S. Lim, S.-Y. Jung, H. Choi, T. Kang, G. Park, C.-W. Yang, J.-G. Choi, G. Ko, J. Shin, I. Yang, J. Nam, H. Sohn, S.-I. Hong, Y. Jeong, S.-W. Choi, C. Choi, H.-S. Shin, J. Lim, D. Youn, S. Nam, J. Lee, M. Ahn, H. Lee, S. Lee, J. Park, K. Gwon, W. Jeong, J. Choi, J. Kim, and K.-W. Jin, "A 176-stacked 512gb 3b/cell 3d-nand flash with 10.8gb/mm2 density with a peripheral circuit under cell array architecture," in *IEEE International Solid-State Circuits Conference (ISSCC)*, vol. 64, 2021, pp. 422–423.

[87] K.-T. Park, S. Nam, D. Kim, P. Kwak, D. Lee, Y.-H. Choi, M.-H. Choi, D.-H. Kwak, D.-H. Kim, M.-S. Kim, H.-W. Park, S.-W. Shim, K.-M. Kang, S.-W. Park, K. Lee, H.-J. Yoon, K. Ko, D.-K. Shim, Y.-L. Ahn, J. Ryu, D. Kim, K. Yun, J. Kwon, S. Shin, D.-S. Byeon, K. Choi, J.-M. Han, K.-H. Kyung, J.-H. Choi, and K. Kim, "Three-Dimensional 128 Gb MLC Vertical nand Flash Memory With 24-WL Stacked Layers and 50 MB/s High-Speed Programming," *IEEE Journal of Solid-State Circuits (JSSC)*, vol. 50, no. 1, pp. 204–213, 2015.

[88] Park, Jisung and Kim, Myungsuk and Chun, Myoungjun and Orosa, Lois and Kim, Jihong and Mutlu, Onur, "Reducing Solid-State Drive Read Latency by Optimizing Read-Retry," in *ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, 2021, p. 702–716.

[89] Pierre-Alain Fouque and Jeffrey Hoffstein and Paul Kirchner and Vadim Lyubashevsky and Thomas Pornin and Thomas Prest and Thomas Ricosset and Gregor Seiler and William Whyte and Zhenfei Zhang, "Falcon: Fast-Fourier Lattice-based Compact Signatures over NTRU," https://api.semanticscholar.org/CorpusID:231637439, 2019, [Online; accessed 06-July-2025].

[90] K. Ramezanpour, P. Ampadu, and W. Diehl, "Scaul: Power side-channel analysis with unsupervised learning," *IEEE Transactions on Computers (TC)*, vol. 69, no. 11, pp. 1626–1638, 2020.

[91] M. Randolph and W. Diehl, "Power side-channel attack analysis: A review of 20 years of study for the layman," *Cryptography*, vol. 4, no. 2, 2020.

[92] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Communications of the ACM (CACM)*, vol. 21, no. 2, pp. 120–126, 1978.

[93] Samsung, "The future of nand technology," https://semiconductor.samsung.com/news-events/tech-blog/the-future-of-nand-technology/, 2019, [Online; accessed 14-June-2025].

[94] Samsung, "Samsung NVMe TCG Opal SSC SEDs PM1733/PM1735 Series," https://csrc.nist.gov/CSRC/media/projects/cryptographic-module-validation-program/documents/security-policies/140sp4238.pdf, 2021, [Online; accessed 20-June-2025].

[95] Samsung, "Download – Tools & Software," https://semiconductor.samsung.com/consumer-storage/support/tools/, 2025, [Online; accessed 06-July-2025].

[96] Sanchez-Avila, C. and Sanchez-Reillol, R., "The Rijndael block cipher (AES proposal) : a comparison with DES," in *IEEE International Carnahan Conference on Security Technology (ICCST)*, 2001, pp. 229–234.

[97] A. Sayakkara, N.-A. Le-Khac, and M. Scanlon, "A survey of electromagnetic side-channel attacks and discussion on their case-progressing potential for digital forensics," *Digital Investigation*, vol. 29, pp. 43–54, 2019.

[98] T. Segment, "The state of personalization 2023," https://gopages.segment.com/rs/667-MPQ-382/images/TS-CNT-Report-The%20State%20of%20Personalization%2023.pdf, 2023, online; accessed 17-June-2025.

[99] Shepherd, Carlton and Akram, Raja Naeem and Markantonakis, Konstantinos, "EmLog: Tamper-Resistant System Logging for Constrained Devices with TEEs," in *Information Security Theory and Practice*, 2018, pp. 75–92.

[100] Y. Shim, M. Kim, M. Chun, J. Park, Y. Kim, and J. Kim, "Exploiting Process Similarity of 3D Flash Memory for High Performance SSDs," in *IEEE/ACM International Symposium on Microarchitecture (MICRO)*, 2019, p. 211–223.

[101] P. Shor, "Algorithms for quantum computation: discrete logarithms and factoring," in *Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, 1994, pp. 124–134.

[102] SK Hynix, "SK hynix PE8111 E1.L NVMe TCG Opal SSC SED," https://csrc.nist.gov/CSRC/media/projects/cryptographic-module-validation-program/documents/security-policies/140sp4410.pdf, 2022, [Online; accessed 25-June-2025].

[103] SK Hynix, "SK hynix 512Gb 176L TLC 3D NAND Internal Waveform Overview," https://library.techinsights.com/analysis-view/IWO-2206-801, 2024, [Online; accessed 19-June-2025].

[104] J. Song, "skip to contentmenu opensearch opennation choice page link [editorial] extraordinary innovation for a more unforgettable world: The story behind samsung's pioneering v-nand memory solution," https://news.samsung.com/global/editorial-extraordinary-innovation-for-a-more-unforgettable-world-the-story-behind-samsungs-pioneering-v-nand-memory-solution, 2021, [Online; accessed 16-June-2025].

[105] State Cryptography Administration, "ublic key cryptographic algorithm SM2 based on elliptic curves," https://www.cnnic.com.cn/ScientificResearch/LeadingEdge/soea/SM2/201312/t20131204_43349.htm, 2013, [Online; accessed 25-June-2025].

[106] State of California Department of Justice, "California Consumer Privacy Act (CCPA)," https://oag.ca.gov/privacy/ccpa, 2024, [Online; accessed 20-June-2025].

[107] Storage Networking Industry Association (SNIA), "Storage Security: Encryption and Key Management," https://www.snia.org/sites/default/files/technical-work/whitepapers/SNIA-Encryption-Key-Management-Tech-Whitepaper.pdf, 2023, [Online; accessed 21-June-2025].

[108] D. Sung, D. Lee, J. Ryu, and J. Lee, "Computation-in-memory in three-dimensional memory device," U.S. Patent 11,461,266 B2, 2022.

[109] Synopsys, "IC Compiler II Implementation User Guide: Version L-2016.03," 2025, [Online; accessed 19-Feburary-2025].

[110] Synopsys, "Synopsys primetime px power analysis solution achieves broad market adoption," https://news.synopsys.com/home?item=123041, 2025, [Online; accessed 16-Feburary-2025].

[111] Synopsys, "Synopsys starrc: Golden signoff parasitic extraction," https://news.synopsys.com/home?item=123041, 2025, [Online; accessed 16-July-2025].

[112] Q. Systems, "How to use self-encrypting drives (seds) on your qnap nas," https://www.qnap.com/en/how-to/tutorial/article/how-to-use-self-encrypting-drives-seds-on-your-qnap-nas, 2021, accessed: July 7, 2025.

[113] T. Tanaka, M. Helm, T. Vali, R. Ghodsi, K. Kawai, J.-K. Park, S. Yamada, F. Pan, Y. Einaga, A. Ghalam, T. Tanzawa, J. Guo, T. Ichikawa, E. Yu, S. Tamada, T. Manabe, J. Kishimoto, Y. Oikawa, Y. Takashima, H. Kuge, M. Morooka, A. Mohammadzadeh, J. Kang, J. Tsai, E. Sirizotti, E. Lee, L. Vu, Y. Liu, H. Choi, K. Cheon, D. Song, D. Shin, J. H. Yun, M. Piccardi, K.-F. Chan, Y. Luthra, D. Srinivasan, S. Deshmukh, K. Kavalipurapu, D. Nguyen, G. Gallo, S. Ramprasad, M. Luo, Q. Tang, M. Incarnati, A. Macerola, L. Pilolli, L. De Santis, M. Rossini, V. Moschiano, G. Santin, B. Tronca, H. Lee, V. Patel, T. Pekny, A. Yip, N. Prabhu, P. Sule, T. Bemalkhedkar, K. Upadhyayula, and C. Jaramillo, "A 768Gb 3b/cell 3D-floating-gate NAND flash memory," in *IEEE International Solid-State Circuits Conference (ISSCC)*, 2016, pp. 142–144.

[114] W. Tang, L. Kiffer, G. Fanti, and A. Juels, "Strategic latency reduction in blockchain peer-to-peer networks," *Proceedings of the ACM on Measurement and Analysis of Computing Systems (POMACS)*, vol. 7, no. 2, pp. 1–33, Jun 2023.

[115] A. Tavakkol, M. Sadrosadati, S. Ghose, J. Kim, Y. Luo, Y. Wang, N. Mansouri Ghiasi, L. Orosa, J. Gómez-Luna, and O. Mutlu, "FLIN: Enabling Fairness and Enhancing Performance in Modern NVMe Solid State Drives," in *ACM/IEEE International Symposium on Computer Architecture (ISCA)*, 2018, pp. 397–410.

[116] Texas Instruments, "AM62P Boot Overview," https://e2e.ti.com/cfs-file/__key/communityserver-discussions-components-files/791/Sitara_5F00_AM62P_5F00_Boot_5F00_Overview.pdf, 2023, [Online; accessed 01-July-2025].

[117] Trusted Computing Group, "Architect's guide: Data security using tcg self-encrypting drive technology," https://trustedcomputinggroup.org/wp-content/uploads/Architects-Guide-Data-Security-Using-TCG-Self-Encrypting-Drive-Technology.pdf, 2013, [Online; accessed 22-July-2024].

[118] Trusted Computing Group, "TCG Storage Security Subsystem Class (SSC): Opal," https://trustedcomputinggroup.org/wp-content/uploads/TCG-Storage-Opal-SSC-v2.30_pub.pdf, 2025, [Online; accessed 20-June-2025].

[119] Vasily Dolmatov, "GOST R 34.12-2015: Block cipher "kuznyechik"," https://datatracker.ietf.org/doc/rfc7801/, 2016, [Online; accessed 24-June-2025].

[120] R. S. Waters and E. E. Swartzlander, "A reduced complexity wallace multiplier reduction," *IEEE Transactions on Computers (TC)*, vol. 59, no. 8, pp. 1134–1137, 2010.

[121] M. Wei, L. M. Grupp, F. E. Spada, and S. Swanson, "Reliably erasing data from flash-based solid state drives," in *USENIX Conference on File and Stroage Technologies (FAST)*, 2011, p. 8.

[122] Z. Xu, T. Mauldin, Z. Yao, S. Pei, T. Wei, and Q. Yang, "A Bus Authentication and Anti-Probing Architecture Extending Hardware Trusted Computing Base Off CPU Chips and Beyond," in *ACM/IEEE International Symposium on Computer Architecture (ISCA)*, 2020, pp. 749–761.

[123] Z. Xu, T. Mauldin, Z. Yao, S. Pei, T. Wei, and Q. Yang, "A Bus Authentication and Anti-Probing Architecture Extending Hardware Trusted Computing Base Off CPU Chips and Beyond," in *ACM/IEEE International Symposium on Computer Architecture (ISCA)*, 2020, pp. 749–761.

[124] Y. Yarom and K. E. Falkner, "Flush+reload: A high resolution, low noise, l3 cache side-channel attack," in *USENIX Security Symposium (USENIX Security)*, 2014, pp. 719–732.

[125] C. Zambelli, L. Crippa, R. Micheloni, and P. Olivo, "Cross-temperature effects of program and read operations in 2d and 3d nand flash memories," in *International Integrated Reliability Workshop (IIRW)*, 2018, pp. 1–4.

[126] C. Zambelli, R. Micheloni, L. Crippa, L. Zuolo, and P. Olivo, "Impact of the nand flash power supply on solid state drives reliability and performance," *IEEE Transactions on Device and Materials Reliability*, vol. 18, no. 2, pp. 247–255, 2018.

[127] C. Zhao, C. Z. Zhao, S. Taylor, and P. R. Chalker, "Review on Non-Volatile Memory with High-k Dielectrics: Flash for Generation Beyond 32 nm," *Nanoscale Research Letters (NRN)*, vol. 7, no. 7, p. 5117–5145, 2014.

[128] M. Zheng, J. Tucek, F. Qin, and M. Lillibridge, "Understanding the robustness of SSDs under power fault," in *USENIX Conference on File and Storage Technologies (FAST)*, 2013, p. 271–284.