第 11 卷 第 3 期
2022 年 5 月
集 成 技 术
JOURNAL OF INTEGRATION TECHNOLOGY
Vol. 11 No. 3
May 2022

引文格式：
陈仁海, 史文燕, 李雅帅, 等. 非易失性内存安全技术综述 [J]. 集成技术, 2022, 11(3): 3-22.
Chen RH, Shi WY, Li YS, et al. A survey on the secure non-volatile memory technology [J]. Journal of Integration Technology, 2022, 11(3): 3-22.

非易失性内存安全技术综述

陈仁海[1,2*] 史文燕[1,2] 李雅帅[1,2] 冯志勇[1,2]

[1]（天津大学深圳研究院 深圳 518000）
[2]（天津大学 天津 300350）

摘 要 大数据应用对内存容量的需求越来越大，而在大数据应用中，以动态随机存储器为内存介质的传统存储器所凸显出来的问题也越来越严重。计算机设计者们开始考虑用非易失性内存去替代传统的动态随机存储器内存。非易失性内存作为非易失的存储介质，不需要动态刷新，因此不会引起大量的能量消耗；此外，非易失性内存的读性能与动态随机存储器相近，且非易失性内存单个存储单元的容量具有较强的可扩展性。但将非易失性内存作为内存集成到现有的计算机系统中，需要解决其安全性问题。传统的动态随机存储器作为内存介质掉电后数据会自动丢失，即数据不会在存储介质中驻留较长时间，而当非易失性内存作为非易失性存储介质时，数据可以保留相对较久的时间。若攻击者获得了非易失性内存存储器的访问权，扫描存储内容，便可以获取内存中的数据，这一安全性问题被定义为数据的“恢复漏洞”。因此，在基于非易失性内存模组的数据中心环境中，如何充分有效地利用非易失性内存，并保证其安全性，成为迫切需要解决的问题。该文从非易失性内存的安全层面出发，对近年来的研究热点及进展进行介绍。首先，该文总结了非易失性内存所面临的主要安全问题，如数据窃取、完整性破坏、数据一致性与崩溃恢复，以及由加解密和完整性保护技术引入而导致的系统性能下降等问题。然后，针对上述各问题，对组合计数器模式加密技术、完整性保护技术扩展的默克尔树、数据一致性与崩溃恢复技术，以及相关优化方案作了详细介绍。最后，对全文进行了总结，并对非易失性内存未来需要进一步关注的问题进行了展望。

关键词 非易失性内存；安全；加解密；完整性；持久化
中图分类号 TP 333.1 **文献标志码** A doi: 10.12146/j.issn.2095-3135.20211001002

收稿日期：2021-10-01 **修回日期**：2021-12-03
基金项目：深圳市科创委学科布局项目（JCYJ20170816093943197）；国家自然科学基金委员会面上项目（62072333）
作者简介：陈仁海（通讯作者），副教授，研究方向为计算机存储系统，E-mail：renhai.chen@tju.edu.cn；史文燕，硕士研究生，研究方向为非易失性内存管理技术；李雅帅，硕士研究生，研究方向为非易失性内存管理技术；冯志勇，教授，研究方向为服务计算。

A Survey on the Secure Non-Volatile Memory Technology

CHEN Renhai[1,2*] SHI Wenyan[1,2] LI Yashuai[1,2] FENG Zhiyong[1,2]

[1](*Shenzhen Research Institute of Tianjin University*, *Shenzhen* 518000, *China*)

[2](*Tianjin University*, *Tianjin* 300350, *China*)

[*]*Corresponding Author*: *renhai.chen@tju.edu.cn*

Abstract Big data applications have an increasing demand for memory capacity, but traditional memory using DRAM as a memory medium has become more and more serious in big data applications. Computer designers began to consider using Non-Volatile Memory (NVM) to replace traditional DRAM memory. As a non-volatile storage medium, NVM does not need to be dynamically refreshed, so it will not cause a large amount of energy consumption; at the same time, the read performance of NVM is similar to that of DRAM, and the capacity of a single NVM storage unit has strong scalability. However, integrating NVM as a memory into an existing computer system needs to solve its security problem. Traditional DRAM, as a memory medium, loses data automatically after power failure, so the data will not stay in the storage medium for a long time, while NVM is a non-volatile storage medium, and the data can be retained in the NVM for a relatively long time. If attackers gain access to the NVM and then scan the contents, they can obtain the data in the memory. This security issue is defined as a "recovery vulnerability" of the data. Therefore, in a data center environment based on NVM modules, how to make full and effective use of NVM and ensure its safety has become an urgent problem to be solved. Starting from the security aspect of NVM, this article summarizes the research hotspots and progress of NVM security in recent years. First, it summarizes the main security issues faced by NVM, such as data theft, integrity damage, data consistency and crash recovery, and system performance degradation caused by the introduction of encryption and decryption and integrity protection technologies. Then, in view of the above problems, the combined counter mode encryption technology, integrity protection technology Bonsai Merkel Tree, data consistency and crash recovery technology and related optimization schemes are introduced in detail. Finally, the full text is summarized, and the issues that need further attention in the future of NVM are prospected.

Keywords Non-Volatile Memory; security; encryption and decryption; integrity; persistency

Funding This work is supported by Shenzhen Science and Technology Innovation Committee (JCYJ20170816093943197) and National Natural Science Foundation of China (62072333)

1 引 言

随着大数据时代的到来，越来越多的应用需要在内存中对数据进行计算与处理，典型的以 Spark 为代表的大规模数据处理引擎需要将数据存储到内存中进行运算。因此，大数据应用对内存容量的需求越来越大，而在大数据应用中，以动态随机存储器(Dynamic Random Access Memory，DRAM)为内存介质的传统存储器所凸显出来的问题越来越严重。首先，DRAM 是易失性的存储介质，数据存储在电容器中，由于晶体管有漏电电流，随着时间的推移，会导致电容上

所存储的电荷数量不足以正确地判别甚至丢失数据。因此，需要动态刷新以保证数据不被损坏，但刷新过程中会导致大量的能量被消耗。其次，存储信息的电容需要足够大的容量，以保证信息在存储器中有足够长的驻留时间。DRAM 单个存储单元的容量已经很难从工艺上进一步优化，这意味着很难提高其单位面积的存储容量，最终导致以 DRAM 为内存模组的存储容量的可扩展性出现瓶颈。由于 DRAM 的限制和约束，计算机设计者们开始考虑用非易失性内存(Non-Volatile Memory，NVM)替代传统的 DRAM 内存。NVM 作为非易失的存储介质不需要动态刷新，因此不会引起大量的能量消耗，其读性能和 DRAM 相近，且单个存储单元的容量具有较强的可扩展性[1-5]。NVM 具有低延时、高密度和低功耗等特性，可有效缓解存储墙的问题[6-8]。此外，NVM 还具有可扩展、掉电快恢复等优势。

由于 NVM 具有传统内存的字节可寻址特性和外存的非易失特性，因此，可同时替代内存(如 DRAM)和外存(如机械式硬盘、基于闪存的固态硬盘等)。这将为未来存储系统的发展提供新的选择，甚至彻底颠覆传统的二级存储结构，给计算系统带来更大的存储容量和更短的访问延时，从而使数据与计算更加紧密，适应于当前越来越多的海量数据应用场景。随着英特尔基于 3D XPoint 的 Optane DIMM 技术的成熟和发展，基于非易失性存储器的内存模组在数据中心逐渐得到推广。英特尔已发布的 Optane DIMM 的单条存储容量可以高达 3 TB，其为数据中心提供了大容量、低功耗和高性能的存储解决方案，同时也为传统的内存架构带来了新的变革。

近年来，在计算机体系结构中，安全和隐私问题得到了极大关注。传统的 DRAM 内存存在许多安全问题，如内存泄露、冷启动攻击、侧信道攻击、重放攻击等。传统内存存在的安全问题在 NVM 中依然存在，由于 NVM 自身的非易失性和耐久性，其面临的攻击方法、攻击效果和防御方法都与传统内存有所不同[9-10]。如 NVM 显然更容易受到冷启动攻击[11]，较短的写入寿命也让其更容易受到恶意程序的磨损攻击等。

原则上，安全计算系统目前定义的可信计算基仅包括处理器，即片上资源是可信的；片外组件包括内存总线和内存模块，都是不可靠的，容易受到被动(窥探)和主动(篡改)攻击。传统的以 DRAM 为内存介质的存储器掉电后数据会自动丢失，即数据不会在存储介质中驻留较长时间。而将 NVM 作为非易失性存储介质时，数据可以在 NVM 中保留较长的时间。若攻击者获得 NVM 存储器的访问权后，扫描存储内容，便可以获取其存储数据，该安全性问题被定义为数据的“恢复漏洞”。恶意程序利用 Meltdown 和 Spectre 获取存储在其他运行在内存中程序的机密信息。总线监听和内存扫描攻击技术给 NVM 内存中敏感数据的机密性带来了严重威胁，攻击者还可通过修改、拼接以及重放 NVM 中内存块的数据，来破坏其完整性。

应用传统的加密技术可保证 NVM 数据的机密性，但传统加密算法通常具有较高的计算复杂度，且 NVM 存储单元的数据写入速度相对较慢，因此，加密算法的引入会严重影响 NVM 的性能，特别是写性能。此外，加密算法(如 AES 对称加密算法)通常会进行多次循环迭代，数据被多次写入内存中，这些特性都急剧地增加了数据的写入次数，导致系统性能严重下降，器件寿命缩短[12-14]。“高效”地利用加密算法是保证非易失性内存安全性的重要基石。原本针对 DRAM 系统提出的计数器模式加密(Counter Mode Encryption，CME)(具有高安全级别和低解密延迟等优势)[15-17]，是目前安全 NVM 系统采用的主流加解密方案[18]，相关工作也大多围绕 CME 技术展开。

应用传统的完整性保护技术，可以保护

NVM 系统数据不受篡改攻击，如攻击者通过访问未经授权的内存空间进行代码重用、代码注入和发起面向数据的攻击。与机密性保护技术相比，目前针对 NVM 的完整性保护技术方案比较固定，主要围绕默克尔树(Merkle Tree，MT)展开。MT 是一个基于迭代的哈希树形结构，具有一定的增量校验功能，即若只修改一部分用户数据，那么只需更改 MT 上受到影响的哈希值，而无须更改全部的哈希值。同时，MT 被认为是片上存储开销最低的安全技术方法，因为只有根需要保存在处理器芯片内，所以与简单的数据块消息验证码(Message Authentication Code，MAC)验证相比，MT 验证具有较大优势。目前，安全 NVM 系统较常用的完整树是扩展的 Merkle Tree(Bonsai Merkel Tree，BMT)和英特尔的软件防护扩展技术(Software Guard Extensions，SGX)，常与计数器加密技术 CME 结合使用。此外，由于 NVM 独特的写入耐久性、较高的写功耗和写延时，目前大多研究工作致力于减轻 BMT 验证计算和更新写入开销。

NVM 提供了在内存中持久托管重要数据的可能性。然而，实现数据持久性需要在编写程序时考虑故障安全，即数据的一致性问题。当发生故障时，为了保证持久化的数据不会损坏，需要采用一定的持久性模型，使数据可以在系统故障后恢复到一致的版本。一般通过约定数据的持久化顺序，以及规定需要原子写入 NVM 的数据组合来实现。截至目前，有许多针对数据一致性的持久性模型，如严格持久性、纪元持久性、缓冲纪元持久性、链持久性和事务持久性。内存持久性模型的写约束限制，不仅降低了并行写入性能，而且带来了较高的写入停顿，极大地影响了系统吞吐量。

除数据一致性外，加解密和完整性验证技术的引入也给 NVM 带来了新的一致性需求，即元数据的一致性问题。对于采用 CME 加密和 BMT 完整性验证的安全 NVM 系统，NVM 内存被划分为 4 个区域，分别保存密文、计数器、消息验证码 MAC 和 BMT 节点。除密文外，其他 3 个区域的数据被统称为安全元数据(与密文区分)。通常为了提高性能，安全内存系统将这些经常访问的安全元数据(计数器、BMT 节点)存储在专用缓存空间中，或直接将它们存储在最后一级缓存中。以内存读取和解密为例，若相应的计数器(在 CME 中)已经缓存，则可以并行执行一次性密码本(One Time Pad，OTP)生成和读取访问，隐藏 OTP 生成延迟，提高解密效率。如果将 BMT 经常访问的树节点缓存在芯片上，当验证过程进行到在片上缓存中找到所需的树节点时，即可完成数据块的完整性验证，因为已经对缓存的树节点进行验证，所以其安全性在芯片上得到了保证，大大降低了完整性验证的开销。然而，CPU 缓存中的元数据可能在系统/电源故障后丢失，导致数据和对应的元数据不一致。例如，在数据持久化到 NVM 中但其计数器还没有存入时发生系统故障，那么系统重启后不一致的计数器会导致数据验证失败；如果在计数器持久化到 NVM 中而数据还没有存入时发生系统故障，那么系统重启后旧的数据无法被新的计数器解密。因此，保持数据和安全元数据之间的一致性，对于 NVM 系统的安全至关重要。但是，实现这种崩溃一致性的成本很高，在严格持久化模型下，一次数据更改，会导致数倍的元数据更新操作(取决于 NVM 容量)。每驱逐一个数据块，所有相关的加密计数器和 BMT 中的树节点(从该节点的父节点一直到根节点)也需要刷新到 NVM 中，这会导致系统性能严重下降，并大大缩短 NVM 器件寿命。如何“高效”地保持数据和元数据的一致性，在新的维度进行有效的系统设计和持久化模型设计，是安全 NVM 系统目前面临的一项重要挑战。此外，NVM 的非易失性也使数据的即时恢复成为可能。在亚马逊的云

系统中，停机成本高达每分钟 200 万美元，IT 停机的平均成本为每分钟 5 600 美元，对于一些对目标有高可用性要求的系统，如银行系统和在线交易，通常需要较快的恢复时间。因此，“快恢复”能力也是目前安全 NVM 系统研究工作的热门方向。

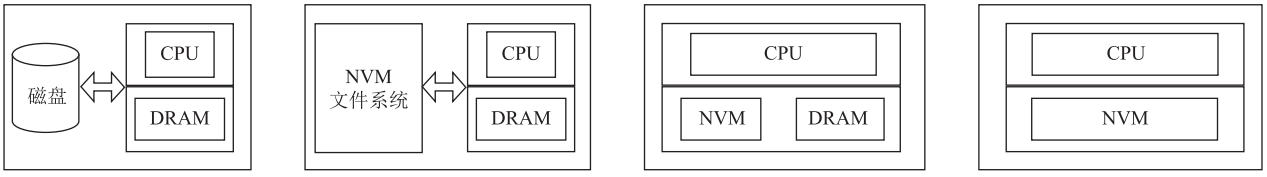
综上所述，NVM 技术的研究热点目前主要分布在以下几个领域：快速加解密技术、完整性验证技术、数据一致性与崩溃恢复技术，以及各种针对由加密和完整性保护技术的引入而造成的 NVM 系统性能严重下降的优化方案。下面将对目前 NVM 各个研究领域中的相关工作展开介绍。

2　非易失性内存的快速加解密技术

NVM 可以直接作为内存，也可以与 DRAM 组合作为混合内存。NVM 参与组成的存储体系架构可能有 4 种情形，如图 1 所示。本文仅讨论 NVM 单独作为内存的非易失性内存安全技术(如图 1(d)所示)。

由于非易失性内存容易受到被动机密性破坏，如通过数据剩余攻击、总线监听攻击等，因此，NVM 必须与内存加密技术相结合。加密可以在内存端或处理器端执行。内存端加密是针对依赖内存作为信任根的内存的独立解决方案。内存端嵌入了一个加密引擎，在数据被写入内存行之前对其进行加密，并在数据被发送到处理器之前进行解密。但是，该方案的保护性较弱，数据仍以明文形式在系统总线上进行通信，无法免于总线监听等攻击导致的机密性泄露。处理器端加密是以处理器芯片为安全边界，使数据免于被动机密性破坏，数据在离开处理器芯片之前被加密，从非易失性内存进入处理器芯片时被解密并验证其完整性。

通过单密钥加解密(如典型的 AES 对称加密算法)直接进行数据加解密，可有效保证数据的安全性，但对 NVM 的读写性能影响较大。一次性密码本的数据保护模式带来的计算开销较低，但需要维护昂贵的密码本。有研究提出一种更加高效的基于一次性密码本的模型来保护 NVM 数据的安全性[18]，该方案的核心思想是通过 AES 这种只需要单密钥的对称算法生成一次性密码本，将需要加密/解密的数据直接和密码本中的数据做相应的运算(如异或运算)，来生成密文/明文。该模型可以利用密码算法提前生成密码本，无须等待数据请求到达后再对数据进行解密，将解密延迟与内存访问延迟重叠，从而利用时间上的并行来降低性能代价。图 2 为拟采用的密码安全体系架构，该架构最早是针对 DRAM 系统加密提出的[19]，也适用于 NVM 系统，称为计数器模式加密技术。该方案首先通过 AES 加密算法将一个计数器作为输入，以生成一次性密码本，当需要加密的数据块到达时，与一次性密码本中的生成密钥按位执行异或操作以获得密文。解密过程是加密过程的逆运算，将密文和密码本中对应的密钥执行异或操作，便可获得明文数据。通过该方法，解密等待时间与存储器访问等待时间重叠，可有效减少因数据安全计算所引



(a) 当前系统结构　(b) NVM 代替磁盘结构　(c) NVM 和 DRAM 混合内存空间结构　(d) NVM 单独作为内存结构

图 1　应用 NVM 的 4 种体系架构

Fig. 1　System architecture options for NVM

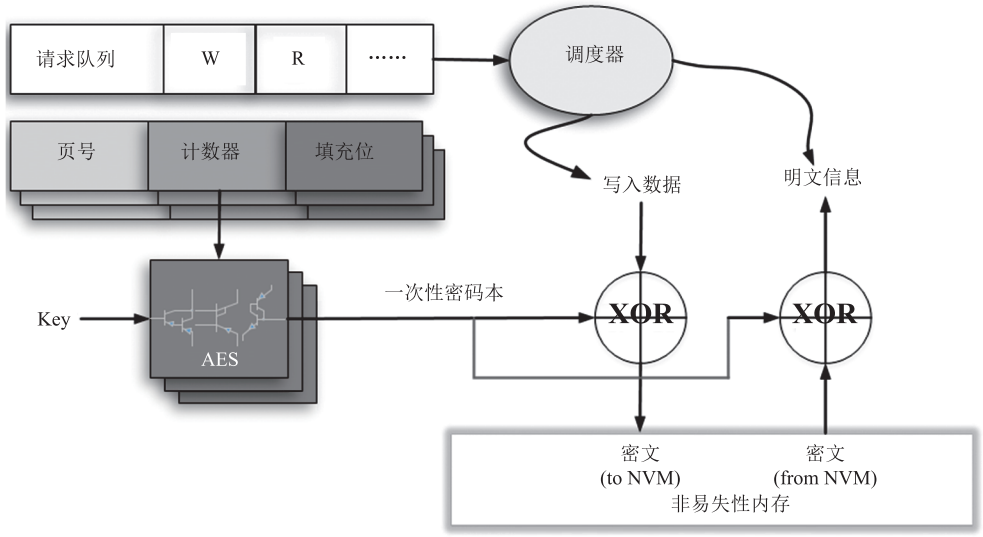


图 2 拟采用的密码安全体系架构

Fig. 2 The proposed cryptographic security architecture

起的计算等待时间。该方案除性能优势外，还可有效抵御各种攻击，包括基于字典的攻击、已知明文攻击和总线监听攻击等攻击手段。

计数器模式加密分为统一计数模式和组合计数模式。统一计数模式是指对所有存储块使用一个全局统一的计数器，每次对写回到 NVM 的数据块进行加密处理的时候，它都会递增，可避免计数器重用。若全局计数器出现溢出，就需要对整个内存重新加密，代价较高，因此，统一计数器通常需要设置较多位数，以防止溢出现象发生，开销较大，并不实用。因而目前多采用组合计数模式。在组合计数器方案中，加密计数器被组织为两部分：包括主要计数器(在同一页的数据块之间共享)和专门针对每个缓存块的次要计数器。该方案通常将 NVM 组织成多个数据页，如以 4 KB 为单位对 NVM 进行页划分。这种设计具有多个优点，首先，在计数器缓存中，仅需按页为单位缓存主要计数器，可有效降低缓存使用代价；其次，当次要计数器溢出时，只需要重置该页所有次计数器并将主要计数器递增，随后用主要计数器重新加密该页数据即可，与统一计数器模式相比，该方案代价较低。此外，主计数器一般设置得足够大(如 64 位)，以确保其在系统生命周期中不会溢出。

然而，仅将传统针对 DRAM 系统的加密技术直接应用于 NVM 系统，并不能达到预期效果。NVM 系统面临的主要挑战之一是如何有效处理写操作，与读取操作相比，NVM 写入具有更高的延迟，功率消耗较大，且器件耐用性不高，因此，需要系统尽可能地减少写入流量，从而提高电源效率、写入带宽及延长器件寿命。将内存加密与 NVM 结合使用是非常具有挑战性的，因为加密扩散效应会加剧 NVM 的写入耐久性问题。Solihin 等[20]为了缓解加密技术带来的性能下降，提出了一种增量加密技术 i-NVMM，该技术将加密单元放入内存端，把频繁访问的内存页面以未加密形式保存在内存中，并以加密形式保护内存的其余部分，然而，未加密内存与片外的通信依然存在安全漏洞。

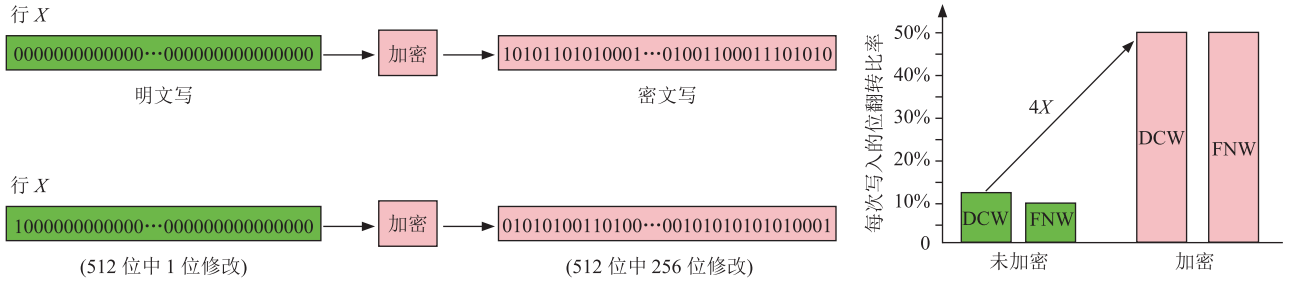
实际上，对主内存执行写操作时，只有少量位被修改，所以早期为了减少对 NVM 的写入，Zhou 等[21]提出了数据比较写入(Data Comparison Write，DCW)技术，即只有缓存行中的修改位被写入 NVM。Cho 等[22]在 DCW 的基础上又提

出了增强型的 N 次翻转写入技术(Flip-N-Write，FNW)，若超过一半的位被修改，则通过反转数据进一步减少对 NVM 的位写入。FNW 限制每次写入时的位翻转次数最多不超过行中位数的一半。通过这些优化，每次进行内存写入操作时，写入的位数平均降至 10%～15%，这些写优化技术对于实现 NVM 的高性能和耐用性至关重要。

然而，加密技术的引入会导致 DCW 和 FNW 等“写优化”技术失效，这是因为所有安全性较好的加密算法都不可避免地遵循雪崩效应[23](或称为扩散效应)，即当明文数据中的单个位发生变化时，会导致加密数据中 50% 的位发生变化(如图 3 所示)。典型的回写平均只会修改缓存行中 12% 的位，而加密算法导致写入 NVM 的位数几乎增加了 4 倍，这种无关位的写入会导致写入功耗显著增加、写入带宽降低和器件寿命缩短，同时也导致了 DCW 和 FNW 等优化技术失效。

针对上述情况，Young 等[24]在传统计数器加密的基础上提出了双计数器加密技术(Dual Counter Encryption，DEUCE)，并观察到，典型的回写操作仅修改若干个字，不必使用新的计数器重新加密行中的所有字。DEUCE 定义了标志位，并利用标志位跟踪记录被修改和未被修改的字，然后使用新的计数器来加密被修改的字，对于未修改的字则继续使用旧的计数器加密(旧的计数器通过屏蔽当前新计数器的几个最低有效位获得)。同样地，在解密时，使用新的计数器解密修改后的字，旧计数器解密缓存行的其余未修改部分，并最后将它们组合起来形成最新的缓存行。

DEUCE 的操作示例如图 4 所示，该加密技术一行有两个计数器，前导计数器(Leading Counter，LCTR)和尾随计数器(Trailing Counter，TCTR)。LCTR 与行计数器相同，



(a) 加密扩散效应示例　　(b) 加密对 DCW 与 FNW 的影响

图 3　加密扩散效应示例及其对 DCW 与 FNW 的影响

Fig. 3　Examples of encryption diffusion effect and its impact on DCW and FNW

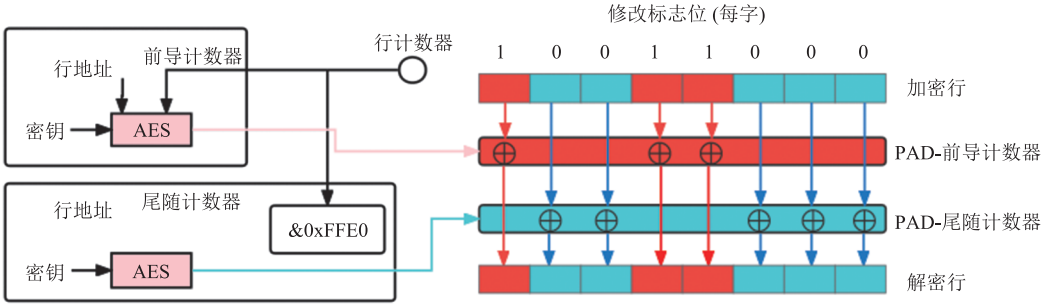


图 4　DEUCE 操作示例

Fig. 4　DEUCE operation example

TCTR 则通过屏蔽 LCTR 的几个最低有效位获得。如屏蔽 LCTR 的两位最低有效位，则每 4 次写入 LCTR 将等于 TCTR，这之间的时间间隔被称为纪元间隔。在纪元间隔之间，TCTR 的值保持不变，而 LCTR 不断增加。每个字都附加一个标志位，指示自纪元开始以来该字是否被修改，当 LCTR 等于 TCTR 时，与该行中所有字相关联的修改标志位都会被复位。LCTR 和 TCTR 都是虚拟计数器，其值由行计数器确定，因此，DEUCE 不需要单独的计数器来实现加密。值得注意的是，纪元间隔和跟踪字长是 DEUCE 的两个关键设计参数，如何设定这两个参数的大小需要视具体情形权衡。纪元间隔太小会导致频繁的全行重新加密，纪元间隔太大会导致系统不断地重新加密不需再修改的字，这些字在纪元刚开始时已被修改过。跟踪的字长则决定了跟踪缓存行修改部分的存储开销和精细度，更细粒度的跟踪会导致更大的存储开销。通过相关评估表明，DEUCE 将写入 NVM 存储位的翻转次数从 50% 降低到 24%，系统性能提高了 27%。

Swami 等[25]在 DEUCE 的基础上提出了 SECRET 加密技术(Smartly Encrypted Energy Efficient，SECRET)，该技术集成了字级重加密技术[24]和基于零的部分写入技术，以减少内存写入流量。字级重加密技术修改了经典计数器的加密方案，为每个字分配单独的计数器，允许以小于缓存线的粒度进行数据加密。基于零的部分写入技术则利用了“在实际工作负载(如 SPEC CPU2006[26])中，写入内存的明文大多为零”这一事实。SECRET 分配一位零标志位来跟踪缓存行中的零字，并将零字的密文保持为最后被加密的状态，从而节省重新加密零字的写入开销。此外，SECRET 还运用能量掩码来过滤加密后的字(即密文)，以执行写优化。对于每个字中每位的能量状态，基于异或的能量掩码可将密文中的高能量状态转换为低能量状态，从而降低缓存线的整体写入能量。表 1 为 SECRET 与块级加密(Block Level Encryption，BLE)[27]和 DEUCE 在能耗降低、延时缩短、寿命提升和内存开销方面的性能对比结果。由表 1 可知，SECRET 明显优于其他两个 NVM 加密解决方案，具有最低的写入能量、最低的写入延迟以及最长的使用寿命。

3 非易失性内存的完整性保护技术

存储器完整性保护的目的在于对数据进行校验，确保系统在没有受到任何外来侵入或篡改的状态下工作。鉴于使用每个内存块的数据和地址的 MAC 值进行完整性验证的方法并不能有效阻止重放攻击，目前针对 NVM 的完整性保护技术主要是围绕 MT 展开的。该技术最先是针对传统 DRAM 系统的安全性提出的[19]。MT 以其增量校验功能和较低的片上存储开销(只有根需要保存在处理器芯片内)优势，成为完整性验证技术的

表 1 BLE、DEUCE、SECRET 性能比对

Table 1 BLE, DEUCE, SECRET performance comparison

NVM	加密技术	能耗降低	延时缩短	寿命提升	内存开销
阻变式存储器	BLE	40%	23%	35%	1.56%
	DEUCE	40%	17%	36%	6.25%
	SECRET	80%	37%	63%	6.25%
多层式存储器	BLE	33%	31%	18%	1.56%
	DEUCE	40%	23%	24%	6.25%
	SECRET	63%	49%	56%	7.84%

注：所有方案均采用基于 AES 的计数器加密技术

首选。

在传统的 MT 中，首先计算每一个数据块的哈希值，得到树的叶子节点；然后向上一级迭代，继续计算其哈希值，直至计算出根节点的哈希值；最后将根节点的哈希值保存在处理器芯片中，从而保证其机密性。在该结构中，只要根节点的哈希值是安全的，那么任何一个数据块的任意位被攻击者篡改，都能被检测到。虽然 MT 能够满足存储器完整性校验的条件，但是系统需要处理较大的计算量和存储空间。因此，Rogers 等[19]在 MT 的基础上提出了 BMT，其结构如图 5 所示。将 BMT 与组合计数器加密模式结合，可以取得较好的加密和完整性验证效果。计数器模式加密的基本安全保证是计数器值不能重复，否则加密可能会失效，为了避免发生计数器重放攻击，计数器必须通过 BMT 保护以防止被篡改。计数器加密与 BMT 结合保证非易失性内存的机密性和完整性，需满足以下 3 个条件：(1) 每个数据块都被其 MAC 值保护；(2) 每个数据块的 MAC 值都包含与数据块对应的计数器和地址的认证；(3) 所有计数器的完整性得到保护。

同样地，考虑到 NVM 有限的写入耐久性、较高的写功耗和写延时，若直接将原本针对 DRAM 系统安全提出的 BMT 应用于 NVM 系统，会出现诸多问题。BMT 用于完整性验证计算和更新消耗的资源较多，其大小取决于树的高度，即 NVM 的容量。例如，当处理器写入数据到 NVM，并更新相应的计数器时，必须更新 MT 中相应的父节点，以反映最近的更改，修改内容从叶子节点向上传播到根节点。图 5 是通用 BMT 方案，该树完全依赖于叶子节点的计数器值建立，是不可并行化的完整树。在进行计数器验证的时候，此种结构虽然可以针对每个级别并行计算，但是当涉及计数器更新时，在下层计算完成之前计算上层的哈希值是不可行的，需要按顺序更新上层，因此，验证计算开销较大。目前，与 NVM 系统完整性保护技术相关的研究工作多致力于减轻 BMT 验证计算和更新写入开销。

英特尔设计的 SGX 树[28]是一种可并行更新的树，其结构如图 6 所示。SGX 树的每个树节点包含 8 个随机数(叶子节点为 8 个计数器)和一个 MAC 值。MAC 是利用存储在芯片中的秘密哈希函数，对当前块中的所有随机数/计数器和来自父块的一个随机数进行哈希计算得到的。当计数器增加时，父节点中的相应随机数也会增加，这使 MAC 值的并行更新成为可能，因为每个块都

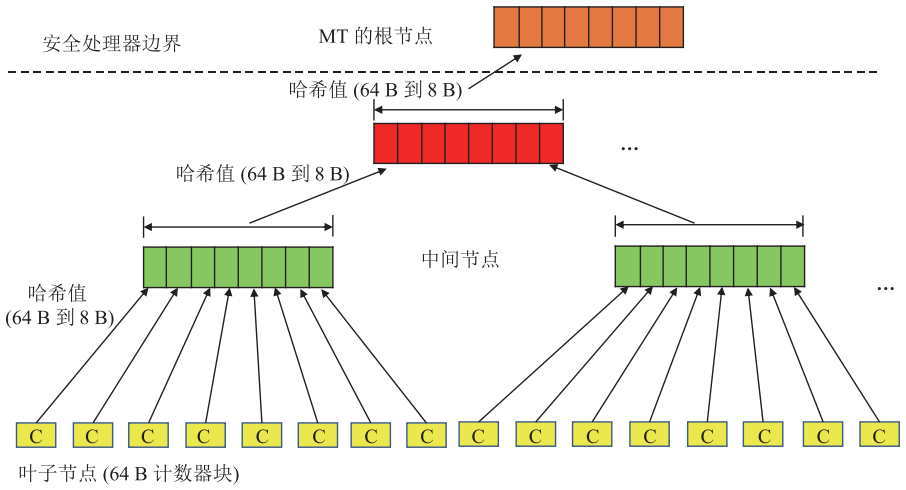


图 5　BMT 结构示意图

Fig. 5　BMT structure sketch map

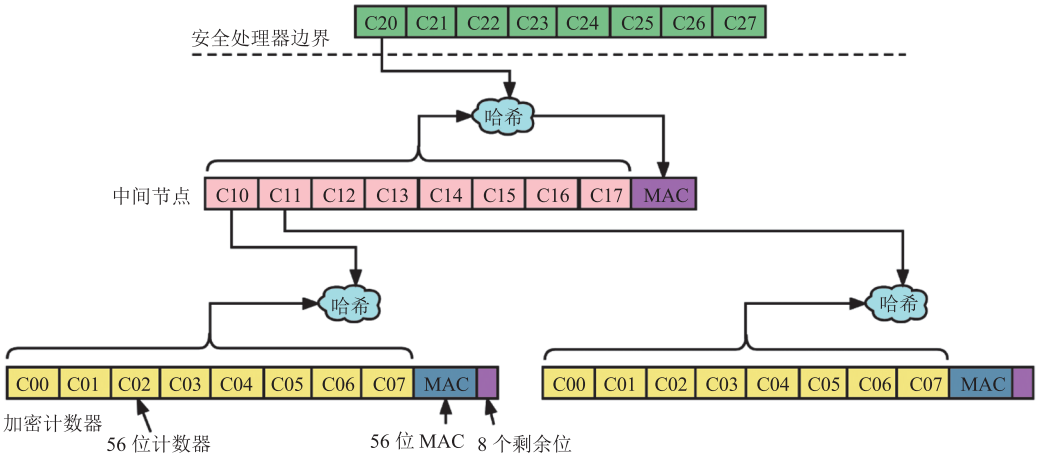


图 6 SGX 树结构图

Fig. 6 SGX tree structure diagram

可以通过增加它们的随机数，并在新的随机数上计算 MAC 来单独更新。因为自身的可并行更新和更低的层次结构(验证更新次数更少)等特点，SGX 树成为目前比较实用的商用方案。

针对传统的基于 DRAM 的安全系统，为了降低 BMT 完整性验证和更新的开销，Gassend 等[29]提出将常用的树节点缓存起来的策略。受此启发，Yang 等[30]针对安全 NVM 提出了一种延迟传播方案 CC-NVM。由于大多数工作负载的局部性，CPU 缓存驱逐的几个相邻数据块很可能共享 BMT 中树节点的相同祖先。因此，若独立更新每个数据块的树节点，就会导致较大的计算冗余，CC-NVM 则消除了这种冗余。当数据从最后一级缓存中被逐出时，元数据更新开始直接在元数据缓存中进行，而当计数器的 MAC 在元数据缓存中命中时，就停止更新，这是因为经过验证和缓存的树节点被认为是安全的。

除了 NVM 系统由于非易失性引起的数据一致性问题[31-33]外，加密和完整性技术的引入也给安全 NVM 系统带来了新的元数据一致性问题。数据一致性[34-35]与元数据一致性问题[36-38]已经成为 NVM 系统必须面对的一项重要挑战，抛开一致性问题讨论完整树验证和更新的开销技术是不可行的。因此，本节仅对 NVM 完整性验证的基本技术及若干优化点进行介绍，与完整性优化相关的技术发展，将在第 4 节中与一致性问题一起讨论。事实上，目前许多致力于缓解完整树验证和计算开销的优秀成果，大多是以 NVM 系统的崩溃一致性为前提的。

4 非易失性内存的安全持久化与崩溃恢复

一般情况下，为了提高性能，安全 NVM 系统会将经常访问的安全元数据(如计数器、BMT 节点等)存储在专用缓存空间中，或直接将它们存储在最后一级缓存上。但是，CPU 缓存是易失性的，而数据与对应的元数据通常是非原子性写入 NVM，因此在系统/电源故障后，缓存数据的丢失可能会导致 NVM 中数据与对应的元数据不一致。如，数据已经写回到 NVM，但是对应的计数器并没有写回，那么系统重启后该数据就会验证失败。同样，如果计数器写回到 NVM 而数据还没有写回时系统崩溃，那么重启后旧的数据无法被新的计数器解密。因此，保证数据与安全元数据之间的一致性，对于安全 NVM 系统来说至关重要。除保证崩溃一致性外，“快恢复”能力也是目前安全 NVM 系统研究工作的重点。NVM 的非易失性使得数据的即时恢复成为

可能，如可实现秒开机。大多数数据中心、云系统、间歇性电源设备甚至是个人计算机，都期望系统能够在电源恢复后立即恢复，这也是 NVM 系统的优势之一。

在严格持久化模型中，实现崩溃一致性的成本很高，因为一次数据的更改，将会导致数倍的元数据更新。每驱逐一个数据块，所有相关的加密计数器和 BMT 中的节点(从该节点的父节点一直到根节点)也需要刷新到 NVM 中，这会导致系统性能严重下降，极大地缩短 NVM 器件寿命。如何“高效”地保持数据和元数据的一致性，从新维度有效地进行系统设计和持久化模型设计，是安全 NVM 系统目前面临的一项重要挑战。

基于硬件的日志持久事务[36-37]是保证数据崩溃一致性的常用机制。它确保在数据更新到位之前，数据日志条目就已经写入 NVM 中的日志区域，以便系统发生故障后，可使用日志将数据恢复到一致版本。计数器原子性方案[38]提出了选择性的计数器原子性概念，它放宽了部分计数器的持久化开销，因为与这些计数器相关联的数据不会立即影响系统的可恢复性。但该方案需要在软件和硬件层上都进行修改，包括编程语言、编译器和内存控制器，并且可能会导致 OTP 的重用。Swami 等[39]通过压缩数据块来确保数据块和相关元数据之间的原子性，数据块、MAC 和计数器被放置在单个缓存行中。Osiris[40]将与数据位于同一位置的纠错码(Error Correct Code，ECC)位作为完整性检查手段，帮助检索计数器，并将 ECC 位与止损机制(计数器每 N 次更新)相结合。通过检查 ECC 位的完整性，只需尝试几次即可恢复每个计数器。该方案成功地将写回策略集成到计数器持久化中，从而减少了 NVM 的写入流量。但该方案无法确定具体丢失的计数器值，所以需要无差别地恢复所有计数器值，这会导致系统的恢复时间过长，甚至达到几小时(NVM 容量是 TB 级别)。此外，Osiris 还依赖于 BMT 对候选计数器值进行最终验证，但是对于用 SGX 树进行完整性验证的安全 NVM 系统，Osiris 并不适用，因为仅恢复加密计数器并不足以重建 SGX 树。SuperMem[41]通过写合并方案减少了用于维护计数器崩溃一致性的写请求，但是它的有效性受到写挂起队列(Write Pending Queue，WPQ)大小和程序位置的限制。

Awad 等[42]提出了一种针对安全 NVM 系统的高性能持久化架构 Triad-NVM。Triad-NVM 强制将多级 BMT 节点保存到 NVM 中，系统崩溃后，Triad-NVM 从叶子节点重建整个 BMT 树，然后将重建的根与存储在芯片上的根进行比较。与将 NVM 系统的安全性和持久性分开讨论的研究不同，该项研究综合考虑了持久性和内存安全，首次讨论了如何持久化地给非易失性内存的持久性区域和非持久性区域提供数据和安全元数据(计数器、MAC 和 BMT)，从整体上分析了持久化、性能、持久化的松散弹性程度以及恢复时间等问题，并给出了优化方案。通过评估表明，与严格持久化技术相比，Triad-NVM 将吞吐量提高了 2 倍左右，同时将 8 TB NVM 的恢复时间控制在 4 s 内，与没有安全元数据持久化的系统恢复时间相比，小了 3 个数量级。

图 7 为 Triad-NVM 的写操作机制示例。WPQ 被认为是属于现代处理器的持久域，到达那里的任何数据都应该是一致的。因此，写操作在被持久化之前，会将其所有相应的更新(计数器、数据、BMT 节点和根)记录到处理器内部的持久性寄存器中，然后设置一个持久性位(称为 READY_BIT)。如果在将持久寄存器中的更新复制到 WPQ 时发生崩溃，那么当系统恢复时，内存控制器会再次尝试将持久寄存器中的更新写入 NVM(或 WPQ)。在将内容从寄存器复制到 WPQ 时，Triad-NVM 可以选择是将计数器或 MT 节点复制到 WPQ，还是仅在缓存中进行更新(如步骤⑧和⑨所示)，一旦脏块从缓存中被逐出，它

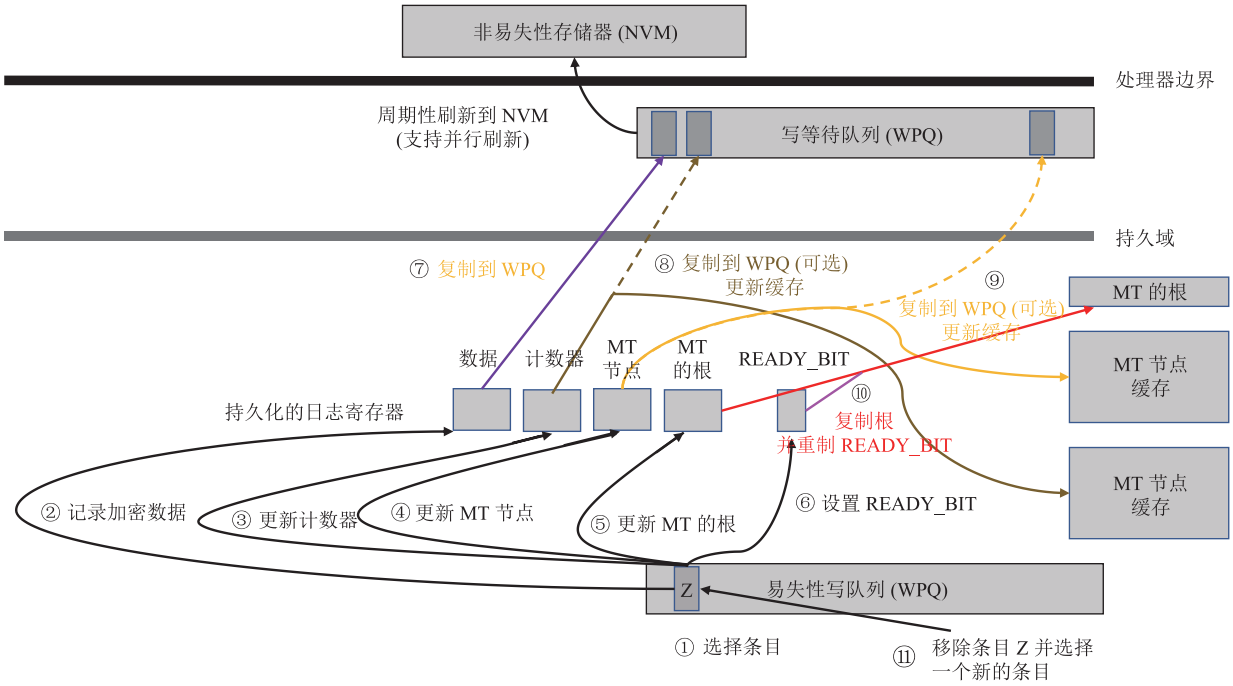


图 7 Triad-NVM 的写操作机制示例

Fig. 7 Example of the write operation mechanism of Triad-NVM

将像往常一样进入 WPQ。持久性寄存器可以利用异步 DRAM 机制来实现，当崩溃发生时，将快速 NVM 寄存器或易失性寄存器刷新到较慢的 NVM 寄存器。

但由于低级 MT 节点的强制持久性，Triad-NVM 会产生 2～4 倍的写入开销，且与 Osiris 一样，Triad-NVM 不支持对 SGX 树的恢复。默克尔树的恢复主要面临两个挑战：(1)可并行的完整性树(如英特尔的 SGX 树[28])由于层间依赖性，无法仅靠恢复叶子节点来重建整棵树，因此，需要非常特殊的处理。(2)容量较大的 NVM 系统(目前商用完整树容量能达到 TB 级别)的恢复时间较长。针对这两个问题，Zubair 等[43]提出了为安全 NVM 系统提供崩溃一致性和快恢复性的方案 Anubis。

Anubis 是一种新颖的纯硬件解决方案，为 NVM 系统提供可恢复性，并确保超短恢复时间，同时产生较小的运行开销。该方案基于一个关键发现：持续跟踪计数器和 MT 在缓存中的块地址，可显著缩短恢复时间。此类缓存中的地址不会频繁更改，仅在未命中时发生更改，因此，在内存中对它们进行跟踪的开销很小。通过跟踪这些地址(如图 8 所示)，在系统崩溃后，只需要重建完整树受影响的部分。此外，Anubis 还支持 SGX 树的可恢复性，称为 ASIT 技术(Anubis for SIT-protected NVM)。对于采用 SGX 树的 NVM 系统，ASIT 镜像复制元数据缓存更新(不仅是地址)到 NVM 上的影子区域，同时采用一个小的 MT 保证该影子区域的完整性，恢复 SGX 树时只需要缓存回影子区域内该树的镜像版本即可。通过相关性能测试表明，与严格持久性相比，Anubis 将平均性能开销从 63% 降低到 3.4%，与 Osiris 开销(1.4%)相近。对于恢复 8 TB 的加密计数器和 BMT，Anubis 仅需 0.03 s，而 Osiris 平均需要 7.8 h。此外，Anubis 的恢复时间并不像其他方案一样随内存大小线性增加，只是安全元数据缓存大小的函数。ASIT 是首个解决一般完整树的恢复时间和 SGX 树的可恢复性

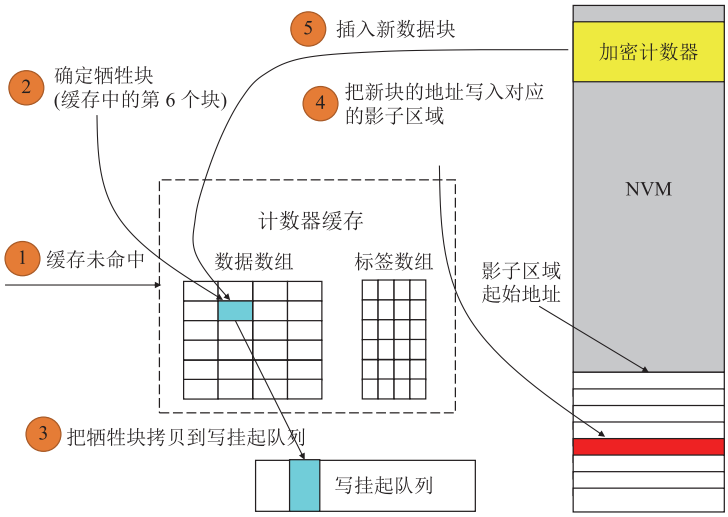


图 8　Anubis 镜像跟踪示例

Fig. 8　Anubis mirror tracking example

问题的方案。

Anubis 虽然实现了完整树的可恢复性，且提供了较短的恢复时间，但每次缓存驱逐都会产生额外的写入，导致 NVM 器件的生命周期缩短了一半，尤其是与 SGX 完整树一起使用时，会产生近 87% 的额外写入开销。Alwadi 等[44]为了弥合安全 NVM 系统的可恢复性和高性能之间的差距，提出了一种新的内存控制器设计 Phoenix。Phoenix 主要针对具有较高的商业价值和安全优势(如抗篡改和重放攻击)的 SGX 完整树。与 ASIT[43]延迟更新元数据缓存但将其更新严格复制到 NVM 的影子区域的方法相比，Phoenix 同样允许使用延迟更新元数据缓存方案，但不严格镜像复制每个计数器更新到 NVM，因为计数器可以通过 Osiris 技术恢复。通过相关性能测试，与不提供可恢复性的加密系统相比，Phoenix 将持久性安全元数据写入开销从 87% 的额外写入(对 ASIT 来说)减少到少于回写方案的写入开销，从而将 NVM 寿命延长 2 倍以上。由于不持久化缓存中计数器块的更新，Phoenix 是首个启用延迟更新元数据缓存却无须记录每个更新的恢复方案。在 ASIT 和 Phoneix 的基础上，Huang 等[45]进一步提出了 SIT 跟踪和恢复方案(SIT Trace and Recovery Scheme，STAR)，该方案删除了对整棵树的额外写入，包括对计数器块和中间树节点的写入。STAR 利用 MAC 字段中未使用的位将父节点的修改存储在其子节点中，仅使用一次原子内存写入就可以将父节点和子节点的修改同时持久化。为了元数据的快速恢复和验证，STAR 还使用 ADR 中的位图行来指示陈旧元数据的位置，并构造一个缓存的 MT 来验证恢复过程的正确性。实验结果表明，与 Anubis 相比，STAR 减少了 92% 的额外内存写入流量，将每时钟指令数(Instruction Per Clock，IPC)开销从 10% 降低到 2%，能耗从 46% 降低到 4%。STAR 恢复具有 4 MB 元数据缓存的系统的安全元数据，仅需要 0.05 s。由于在系统崩溃后系统通常需要 10～100 s 进行自检，因此，该元数据可以忽略不计。此外，STAR 中的恢复时间与元数据缓存中脏元数据的数量成正比，而不是与 NVM 或缓存大小成正比，这为更大的 NVM 和元数据缓存大小提供了适用性。

Lei 等[46]提出了一种专门针对由 SGX 风格完整性树(SGX-style Integrity Tree，SIT)保护的

NVM 的持久化方案(Persistency Solution for SIT-protected NVM，PSIT)(如图 9 所示)。PSIT 采用惰性更新技术，该技术保证，元数据缓存中 SIT 节点的任意最新随机数都能与 NVM 中其对应的子节点相匹配，从而保证系统崩溃后 SIT 节点的恢复能力。基于 SIT 节点的恢复能力，不必立即写回元数据缓存中被更新的元数据。PSIT 使用一个受限的写回型元数据策略，该策略仅当元数据缓存中 SIT 节点的值更新到 N 的倍数时，才会被写回 NVM，这极大地减少了安全元数据的持久化开销，提升了系统性能。PSIT 通过一个地址跟踪器对在缓存中丢失的元数据地址进行跟踪，以加速恢复过程。PSIT 还使用 SGX 树和一个侧 BMT 混合的完整树，以保证整个系统不受完整性破坏，其中，SGX 树保护整个内存的完整性，侧 BMT 保护元数据缓存中已被更新，但还未持久化到 NVM 的块的完整性。侧 BMT 的根被保存在片上持久寄存器中，并总是代表着元数据缓存的最新更新状态。与前述 ASIT 技术相比，PSIT 技术提升了 18% 的系统性能，降低了 47% 的写入流量，同时提供了一个可接受的系统恢复时间。

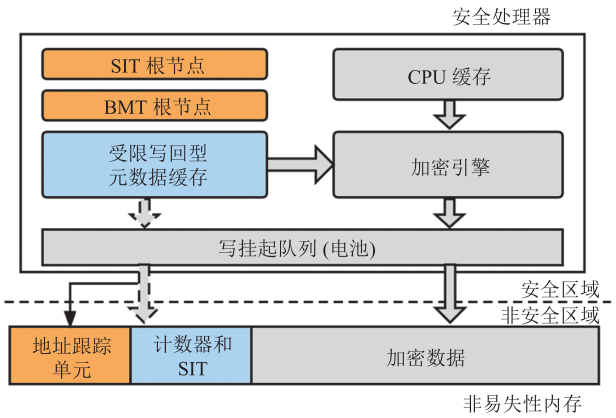


图 9 PSIT 架构示意图

Fig. 9 Schematic diagram of PSIT architecture

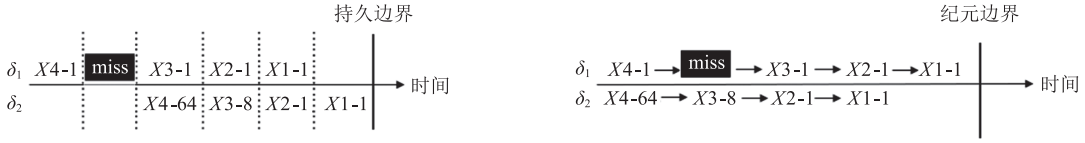
由于 BMT 的叶到根更新必须遵循持久化顺序，否则当完整性系统恢复时，崩溃恢复可能会触发验证失败。但若遵循持久化顺序，在严格持久性模型下，崩溃恢复的开销会导致系统性能下降 30 倍。针对该情况，Freij 等[47]提出了一种针对完整树更新的持久级并行方案。该方案在确保系统正确崩溃恢复的前提下，可提高 BMT 更新的并行水平，从而显著提升系统性能。该方案提出了适用于内存严格持久模型的流水线更新机制(如图 10 所示)和适用于内存纪元持久模型的乱序更新与更新合并机制(如图 11 和 12 所示)。其中持久化 δ_1 和持久化 δ_2 分别代表两个 BMT 持久化更新，变量 X 代表 BMT 节点，箭头方向代表更新路径。图 10(a)中，持久化 δ_1 和持久化 δ_2 的更新路径无序，但对 BMT 根的更新保持持久化顺序，该方法虽然能够显著提升系统性能，但是会造成写后写(Write-After-Write，WAW)的危害。图 10(b)中的祖先节点有序流水线更新方法可有效避免 WAW 并在一定程度上提高持久化的并行性。该方法只有当较旧的持久化已完成对同一级别 BMT 节点的更新时，才允许较新的持久性更新某个级别的 BMT。图 11(a)中，当持久化 δ_1 尝试更新 BMT 时，在 X4-1 节点上发生缓存未命中，这会在有序流水线 BMT 更新中引入气泡，从而导致持久化 δ_2 被延迟，使其无法在 δ_1 更新 X4-1 之前更新 X4-64。图 11(b)中使用无序更新技术，δ_1 更新和 δ_2 更新可以同时进行，δ_2 不会被 δ_1 必须等待的缓存未命中延迟。无序更新技术相对于有序流水线更新技术可以实现更高程度的持久级并行。通过相关性能测试表明，与顺序更新机制相比，流水线更新机制将系统性能提高了 3.4 倍，无序更新和合并更新机制将系统性能提高了 5.99 倍，这些优化显著缩短了更新完整性树根所需的时间，并为安全 NVM 的实用化奠定了基础。

Chen 等[48]提出了一种 CacheTree 技术。该技术通过在元数据缓存上构建额外的 MT，验证易失性缓存内容，使系统能够采用回写策略，消除持久化元数据导致的对 NVM 的频繁写入，极大地

(a) BMT 乱序更新和根有序更新　　(b) 祖先节点有序(包含根节点)的流水线更新

图 10　BMT 并行化更新示例

Fig. 10　BMT parallelized update example



(a) 有序流水线更新　　(b) 无序更新

图 11　流水线技术和无序更新技术示例

Fig. 11　Examples of pipelining and out-of-order update techniques

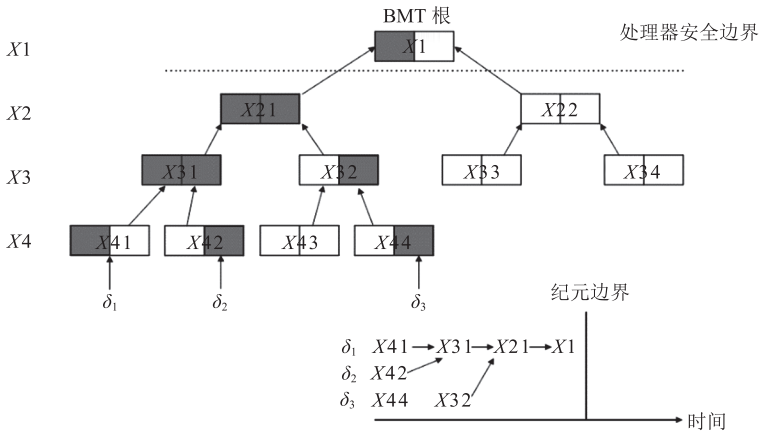


图 12　合并更新示例

Fig. 12　Example of merged update

降低了持久化消息验证码和持久化 MT 的开销。

图 13 为 CacheTree 的设计概览。中间部分是传统的安全增强组件，包括内存控制器、AES 加密引擎、MAC 缓存、计数器 Cache、MT 缓存和一个用于保存 BMT 根的安全非易失性寄存器。上层部分是启用 CacheTree 的块。该技术分别使用 Osiris[40]技术更新计数器缓存、MACTree 更新 MAC 缓存、HNodeTree 更新 BMT。CacheTree 用不到 0.5% 的存储开销，实现了高达 20.1% 的性能提升、44.3% 的生命周期延长，以及 43.7% 的能耗缩减。

基于硬件日志的持久事务[36-37]是保证数据崩溃一致性的常用机制。它确保在数据更新到位之前，数据日志条目被写入 NVM 中的日志区域，以便在系统故障后，利用日志将数据恢复到一致版本。ATOM[36]是针对 NVM 持久化方案中较为先进的硬件实现，该方案基于日志的持久化事务。在此基础上，Lei 等[49]提出了一种共享计数器和延迟计数器持久化的 CCAE 技术，该技术主要包括两部分：用于日志加密的共享计数器优化和用于数据加密的延迟计数器持久化。

CCAE 的提出主要源自两个关键发现：(1) 由于写入附加功能，事务中使用的所有日志条目在被回收之前都具有相同的加密次数，使共享计

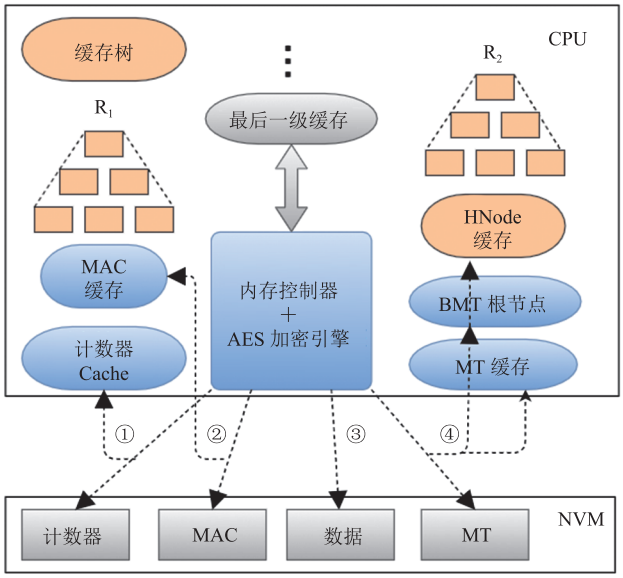


图 13 CacheTree 结构概览

Fig. 13 Overview of the CacheTree structure

数器的优化成为可能。共享计数器的优化方案是在日志分配的粒度上对一个日志区使用全局加密计数器，且该日志区中的所有日志缓存行都使用共享计数器进行加密/解密，该技术大大减少了加密日志计数器的数量，从而降低了存储开销。(2)与未提交数据块相关联的计数器只需要恢复到崩溃前较新的值，以避免 OTP 重用即可。在系统恢复过程中，未提交的事务中的数据块将被丢弃，所以不需要保证其相关联的计数器恢复到准确的值。因此，CCAE 在事务执行期间，会吸纳并延迟所有由计数器持久化引起的写请求。在事务提交前，事务中更新的计数器不会持久保存到 NVM 中，当一个事务发出提交指令时，所有与该事务对应的被更新的计数器缓存行才会被写回 NVM。当脏数据高速缓存行和所有与此事务关联的计数器高速缓存行都被持久化时，表明事务被成功提交。延迟计数器持久化方案有效地减少了事务中与数据块关联计数器的写请求。

图 14 为 CCAE 的硬件架构。它主要包括两个模块：日志管理模块与数据加密模块。日志管理部分与 ATOM 相同，加密模块则采用传统的计数器模式加密方案(Counter Mode Encryption，CME)。通过相关性能测试表明，CCAE 减少了 67% 的由计数器引起的 NVM 写入流量，提升了 14% 的系统性能，降低了 35% 的 NVM 能耗。

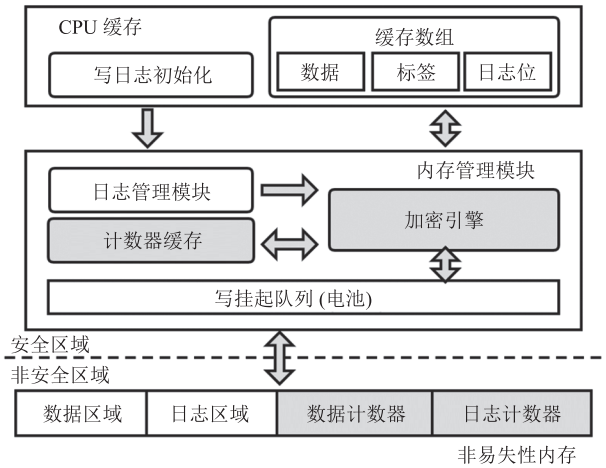


图 14 CCAE 硬件结构图

Fig. 14 CCAE hardware structure diagram

5 总结与展望

本文从 NVM 的安全层面出发，针对 NVM 可能遇到的数据窃取、完整性破坏、数据一致性等问题，介绍了现有的解决方案。针对数据窃取给 NVM 系统带来的机密性破坏，本文首先介绍了组合计数器加密技术，该技术由于高安全级别和低解密延迟等优势，成为 NVM 的主流加解密方案。随后介绍了 DEUCE、SECRET 等针对计数器加密的优化技术，这些技术旨在解决由加密的扩散效应引起的写放大。针对 NVM 系统的完整性破坏方面，本文介绍了 BMT 与 SGX 树，还详细介绍了为减轻完整树验证和更新开销而提出的 CC-NVM 方案。对于数据一致性和崩溃恢复问题，本文先分析了加解密及完整性验证技术给 NVM 系统带来的数据和元数据一致性问题，随后详细介绍了 Osiris、Triad-NVM、Anubis、Phoniex、STAR、PIST、PLP、CacheTree、CCAE 等基于硬件的一致性与崩溃恢复方案，

这些方案有效地缓解了 NVM 的写入流量并保证了数据的一致性。针对 8 TB 的加密计数器和 BMT，Anubis 实现了仅 0.03 s 的恢复时间。

随着 NVM 技术的发展和应用，其暴露出的安全问题受到了更多的关注，设计者需要从整个计算系统出发，自上而下地关注其安全性，包括硬件安全、操作系统安全、编程模型安全等。针对 NVM 加密技术已经做了大量研究，但仍存在许多未解决的问题，加密技术所引起的扩散效应仍严重地影响 NVM 的寿命，未来需要提出更加高效的解决方案，以应对此扩散效应。在针对 NVM 完整性保护技术中，基于完整树的完整性验证和更新的开销仍然较大，降低此类开销的技术也是未来的一个重要研究方向。此外，若 NVM 系统崩溃后受到了不可恢复的数据破坏，现有的恢复方案无法识别受到攻击的具体位置。如 Osiris 恢复方案虽然能通过恢复叶子节点来重建整棵 BMT，但是若重建后的根节点与芯片上保存的根节点对比不一致，Osiris 技术无法识别具体是哪一部分数据遭到了攻击。同样地，当攻击者替换位图行或安全元数据时，STAR 技术可以检测到攻击发生，并使系统恢复，但无法定位受到攻击的数据块。鉴于准确定位被攻击的数据位置对于系统防护是非常重要的，未来的恢复方案需要考虑这一点。在操作系统层面，若 NVM 集成了外存的存储功能，应用程序就可以绕过文件权限管理机制，通过 Load/Store 指令接口直接访问 NVM，这就有可能导致恶意程序破坏 NVM 区域。访问机制的变化需要变更现有的存储软件栈，并重新设计权限管理等安全角色。在编程模型安全方面，较受关注的是程序的可移植性和迁移成本问题，即应用程序从易失的计算系统迁移到受安全技术保护的非易失系统，如何使迁移的成本更小也是未来的工作重点。除传统的加解密和完整性保护技术外，一些辅助增强 NVM 系统安全性的工作如地址随机化技术、减少内存暴露时间等，也是确保系统安全的重要研究方向。

6　结束语

随着云计算、大数据、人工智能、物联网、高性能计算等新技术在各行业得到越来越多的应用，这意味着人类已经进入了一个全新的数据时代。新的数据时代不仅产生了大量数据，而且对数据的计算和存储提出了更高的要求，传统的内存架构迫切需要变革，以适应不断增长的数据需求。NVM 由于其非易失性、字节寻址能力和空闲低功耗等优点，成为下一代内存架构的有力候选。本文从 NVM 的安全层面出发，综述了近年来针对 NVM 安全性的研究热点与研究进展，如快速加解密、完整性验证、一致性和崩溃恢复及相关性能的优化技术。虽然目前 NVM 的发展还面临一些问题，但是，随着研究的深入、技术的推进和其自身的独特优势，相信 NVM 会给未来的存储架构提供新的思路和选择。

参 考 文 献

[1] Boukhobza J, Rubini S, Chen RH, et al. Emerging NVM: a survey on architectural integration and research challenges [J]. ACM Transactions on Design Automation of Electronic Systems, 2018, 23(2): 1-32.

[2] Liu W, Wu K, Liu JL, et al. Performance evaluation and modeling of HPC I/O on non-volatile memory [C] // Proceedings of the 2017 International Conference on Networking, Architecture, and Storage, 2017: 1-10.

[3] Xia F, Jiang D, Xiong J, et al. HiKV: a hybrid index key-value store for DRAM-NVM memory systems [C] // Proceedings of the 2017 USENIX Annual Technical Conference, 2017: 349-362.

[4] Chen RH, Yi W, Hu JT, et al. Unified non-volatile memory and NAND flash memory architecture in smartphones [C] // Proceedings of the 20th Asia

and South Pacific Design Automation Conference, 2015: 340-345.

[5] Kim J, Hong AJ, Kim SM, et al. A stacked memory device on logic 3D technology for ultra-high-density data storage [J]. Nanotechnology, 2011, 22(25): 254006.

[6] Chen RH, Wang Y, Hu JT, et al. vFlash: virtualized flash for optimizing the I/O performance in mobile devices [J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2016, 36(7): 1203-1214.

[7] Arulraj J, Pavlo A. How to build a non-volatile memory database management system [C] // Proceedings of the 2017 ACM International Conference on Management of Data, 2017: 1753-1758.

[8] Korgaonkar K, Bhati I, Liu HC, et al. Density tradeoffs of non-volatile memory as a replacement for SRAM based last level cache [C] // Proceedings of the 2018 ACM/IEEE International Symposium on Computer Architecture, 2018: 315-327.

[9] 李月, 王芳. 基于 NVM 的存储安全综述 [J]. 计算机科学, 2018, 45(7): 53-60
Li Y, Wang F. Survey on storage security of emerging non-volatile memory [J]. Computer Science, 2018, 45(7): 53-60.

[10] 徐远超, 闫俊峰, 万虎, 等. 新型非易失存储的安全与隐私问题研究综述 [J]. 计算机研究与发展, 2016, 53(9): 1930-1942.
Xu YC, Yan JF, Wan H, et al. A survey on security and privacy of emerging non-volatile memory [J]. Journal of Computer Research and Development, 2016, 53(9): 1930-1942.

[11] 杨阳, 关志, 陈钟. 冷启动攻击研究综述 [J]. 计算机应用研究, 2015, 32(10): 2886-2890＋2900.
Yang Y, Guan Z, Chen Z. Survey of cold boot attack [J]. Application Research of Computers, 2015, 32(10): 2886-2890＋2900.

[12] Yue JH, Zhu YF. Accelerating write by exploiting PCM asymmetries [C] // Proceedings of the 2013 IEEE 19th International Symposium on High Performance Computer Architecture, 2013: 282-293.

[13] Jiang L, Zhao B, Yang J, et al. A low power and reliable charge pump design for phase change memories [C] // Proceedings of the 2014 ACM/IEEE 41st International Symposium on Computer Architecture, 2014: 397-408.

[14] Schechter S, Loh GH, Strauss K, et al. Use ECP, not ECC, for hard failures in resistive memories [C] // Proceedings of the 37th Annual International Symposium on Computer architecture, 2010: 141-152.

[15] Suh GE, Clarke D, Gasend B, et al. Efficient memory integrity verification and encryption for secure processors [C] // Proceedings of the 36th Annual IEEE/ACM International Symposium on Microarchitecture, 2003: 339-350.

[16] Lipmaa H, Rogaway P, Wagner D. Comments to NIST concerning AES-modes of operations: CTR-mode encryption [C] // Proceedings of the National Institute of Standards and Technologies, 2000: 1-4.

[17] Yan C, Englender D, Prvulovic M, et al. Improving cost, performance, and security of memory encryption and authentication [C] // Proceedings of the 33rd Annual International Symposium on Computer Architecture , 2006: 179-190.

[18] Swami S, Mohanram K. ACME: advanced counter mode encryption for secure non-volatile memories [C] // Proceedings of the 2018 the 55th Annual Design Automation Conference, 2018: 1-6.

[19] Rogers B, Chhabra S, Prvulovic M, et al. Using address independent seed encryption and Bonsai Merkle trees to make secure processors OS- and performance-friendly [C] // Proceedings of the 40th Annual IEEE/ACM International Symposium on Microarchitecture, 2007: 183-196.

[20] Chhabra S, Solihin Y. i-NVMM: a secure non-volatile main memory system with incremental encryption [C] // Proceedings of the 2011 38th Annual International Symposium on Computer

Architecture, 2011: 177-188.

[21] Zhou P, Zhao B, Yang J, et al. A durable and energy efficient main memory using phase change memory technology [J]. ACM SIGARCH Computer Architecture News, 2009, 37(3): 14-23.

[22] Cho S, Lee H. Flip-N-Write: a simple deterministic technique to improve PRAM write performance, energy and endurance [C] // Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture, 2009: 347-357.

[23] Tavares SE, Webster AF. On the design of S-Boxes [C] // Proceedings of the Lecture Notes in Computer Sciences 218 on Advances in Cryptology, 1985: 523-534.

[24] Young V, Nair PJ, Qureshi MK. DEUCE: write-efficient encryption for non-volatile memories [J]. ACM SIGPLAN Notices, 2015, 50(4): 33-44.

[25] Swami S, Rakshit J, Mohanram K. SECRET: Smartly EnCRypted Energy EfficienT non-volatile memories [C] // Proceedings of the 2016 53nd ACM/EDAC/IEEE Design Automation Conference, 2016: 1-6.

[26] Henning JL. SPEC CPU2006 benchmark descriptions [J]. ACM SIGARCH Computer Architecture News, 2006, 34(4): 1-17.

[27] Kong JF, Zhou HY. Improving privacy and lifetime of PCM-based main memory [C] // Proceedings of the 2010 IEEE/IFIP International Conference on Dependable Systems & Networks, 2010: 333-342.

[28] Fuko TA. Intel Software Guard Extensions (SGX) explained [C] // Proceedings of the 9th International Conference on Computer and Systems Engineering, 2019: 1-4.

[29] Gassend B, Suh GE, Clarke D, et al. Caches and hash trees for efficient memory integrity verification [C] // Proceedings of the Ninth International Symposium on High-Performance Computer Architecture, 2003: 295-295.

[30] Yang F, Lu YY, Chen YM, et al. No compromises: secure NVM with crash consistency, write-efficiency and high-performance [C] // Proceedings of the 2019 56th ACM/IEEE Design Automation Conference, 2019: 1-6.

[31] Pelley S, Chen PM, Wenisch TF. Memory persistency [C] // Proceedings of the 2014 ACM/IEEE 41st International Symposium on Computer Architecture News, 2014: 265-276.

[32] Ipek E, Condit J, Lee B, et al. Better I/O through byte-addressable, persistent memory [C] // Proceedings of the Association for Computing Machinery, 2014: 133-146.

[33] Joshi A, Nagarajan V, Cintra M, et al. Efficient persist barriers for multicores [C] // Proceedings of the 48th International Symposium on Microarchitecture, 2015: 660-671.

[34] Kolli A, Pelley S, Saidi A, et al. High-performance transactions for persistent memories [J]. Computer Architecture News, 2016, 44(2): 399-411.

[35] Volos H, Tack AJ, Swift MM. Mnemosyne: lightweight persistent memory [C] // Proceedings of the International Conference on Architectural Support for Programming Languages and Operating Systems, 2011: 91-104.

[36] Joshi A, Nagarajan V, Viglas S, et al. ATOM: atomic durability in non-volatile memory through hardware logging [C] // Proceedings of the 2017 IEEE International Symposium on High Performance Computer Architecture, 2017: 361-372.

[37] Wei XL, Feng D, Tong W, et al. MorLog: morphable hardware logging for atomic persistence in non-volatile main memory [C] // Proceedings of the 2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture, 2020: 610-623.

[38] Liu S, Kolli A, Ren J, et al. Crash consistency in encrypted non-volatile main memory systems [C] // Proceedings of the 2018 IEEE International Symposium on High Performance Computer

Architecture, 2018: 310-323.

[39] Swami S, Mohanram K. ARSENAL: architecture for secure non-volatile memories [J]. IEEE Computer Architecture Letters, 2018, 17(2): 192-196.

[40] Ye M, Hughes C, Awad A. Osiris: a low-cost mechanism to enable restoration of secure non-volatile memories [C] // Proceedings of the 2018 51st Annual IEEE/ACM International Symposium on Microarchitecture, 2018: 403-415.

[41] Zuo P, Hua Y, Xie Y. SuperMem: enabling application-transparent secure persistent memory with low overheads [C] // Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture, 2019: 479-492.

[42] Awad A, Ye M, Solihin Y, et al. Triad-NVM: persistency for integrity-protected and encrypted non-volatile memories [C] // Proceedings of the 46th International Symposium on Computer Architecture, 2019: 104-115.

[43] Zubair KA, Awad A. Anubis: ultra-low overhead and recovery time for secure non-volatile memories [C] // Proceedings of the 46th International Symposium on Computer Architecture, 2019: 157-168.

[44] Alwadi M, Zubair K, Mohaisen D, et al. Phoenix: towards ultra-low overhead, recoverable, and persistently secure NVM [J]. IEEE Transactions on Dependable and Secure Computing, 2020, 19(2): 1049-1063.

[45] Huang JM, Hua Y. A write-friendly and fast-recovery scheme for security metadata in non-volatile memories [C] // Proceedings of the 2021 IEEE International Symposium on High-Performance Computer Architecture, 2021: 359-370.

[46] Lei MY, Wang F, Feng D, et al. An efficient persistency and recovery mechanism for SGX-style integrity tree in secure NVM [C] // Proceedings of the 2020 Design, Automation & Test in Europe Conference & Exhibition, 2020: 702-707.

[47] Freij A, Yuan SG, Zhou HY, et al. Persist level parallelism: streamlining integrity tree updates for secure persistent memory [C] // Proceedings of the 2020 53rd Annual IEEE/ACM International Symposium on Microarchitecture, 2020: 14-27.

[48] Chen ZG, Zhang YT, Xiao N. CacheTree: reducing integrity verification overhead of secure non-volatile memories [J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2020, 40(7): 1340-1353.

[49] Lei MY, Wang F, Feng D, et al. Crash-consistency-aware encryption for non-volatile memories [C] // Proceedings of the 50th International Conference on Parallel Processing, 2021: 1-10.