STM JOURNALS

Recent Trends in
Parallel Computing

ISSN: 2393-8749
Volume 12, Issue 2, 2025
May–August
DOI (Journal): 10.37591/RTPC

https://journals.stmjournals.com/rtpc

Review RTPC

A Review of Blocking Side-Channel Threats in Parallel Cloud Systems

Yamuna Mundru[1,*], Atti Manga Devi[2], Manas Kumar Yogi[3]

Abstract

Side-channel attacks (SCAs) pose a critical security threat to parallel computing systems, particularly in shared cloud environments where multi-tenancy and resource contention create exploitable vulnerabilities. This study presents a comprehensive review of SCAs in parallel architectures, analyzing attack vectors such as cache-based exploits (e.g., Prime + Probe, Flush + Reload), timing attacks, power analysis, and network-based covert channels. We examine real-world cases including Spectre and Meltdown vulnerabilities that exposed fundamental flaws in speculative execution across modern CPUs, along with cloud-specific risks in virtualized environments. The study systematically evaluates detection methodologies, including machine learning-based anomaly detection, hardware performance counter (HPC) monitoring, and formal verification techniques for algorithmic resilience. We then assess mitigation strategies spanning hardware isolation (Intel SGX, AMD SEV), software obfuscation (noise injection), architectural enhancements (constant-time execution), and policy-based controls. Key challenges in balancing security with performance overhead are discussed, alongside emerging threats from quantum computing and distributed architectures. Our analysis reveals that while current defenses reduce attack surfaces, gaps remain in standardized benchmarking, cross-layer protection for heterogeneous systems, and scalable solutions for exascale computing. The study concludes with research directions advocating for: (1) hardware-software co-design to optimize security-performance tradeoffs, (2) quantum-resistant parallel architectures, and (3) unified evaluation frameworks for SCA resilience. This work provides practitioners and researchers with a structured understanding of SCA risks in parallel systems and a roadmap for developing next-generation secure computing infrastructures.

Keywords: Side-channel attacks (SCAs), parallel computing, cloud computing, vulnerabilities, cache, mitigation

***Author for Correspondence**
Yamuna Mundru
E-mail: yamunamundru@gmail.com

[1]Assistant Professor, Department of Computer Science and Engineering (Artificial Intelligence and Machine Learning) (CSE-AI & ML), Pragati Engineering College (A), Surampalem, Andhra Pradesh, India
[2]Assistant Professor, Department of Information Technology, Pragati Engineering College (A), Surampalem, Andhra Pradesh, India
[3]Assistant Professor, Department of Computer Science and Engineering (CSE), Pragati Engineering College (A), Surampalem, Andhra Pradesh, India

Received Date: April 15, 2025
Accepted Date: May 06, 2025
Published Date: June 4, 2025

Citation: Yamuna Mundru, Atti Manga Devi, Manas Kumar Yogi. A Review of Blocking Side-Channel Threats in Parallel Cloud Systems. Recent Trends in Parallel Computing. 2025; 12(2): 15–25p.

INTRODUCTION

Side-channel attacks (SCAs) pose a significant security risk in cloud-based parallel computing systems by exploiting indirect information leaks rather than traditional software vulnerabilities. These attacks leverage physical or behavioral characteristics such as power consumption, timing variations, cache access patterns, or electromagnetic emissions, to extract sensitive data from co-located virtual machines (VMs) or containers [1]. With the growing adoption of multi-tenant cloud infrastructures for high-performance computing (HPC), big data processing, and AI workloads, SCAs have become a critical concern. For instance, attacks like Spectre and Meltdown demonstrated how speculative execution in CPUs could leak data across VM boundaries, affecting millions of cloud servers.

Why Parallel Computing is Vulnerable

Parallel computing environments, particularly those running on shared cloud platforms, are highly susceptible to SCAs due to [2]:

1. *Resource sharing:* Multi-core CPUs, GPUs, and caches are shared among users, enabling attackers to infer activity from neighboring tasks.
2. *Virtualization overheads:* Hypervisors and container orchestration (e.g., Kubernetes) introduce timing variations that can be measured.
3. *Distributed communication:* Message-passing interfaces (e.g., MPI) and RDMA networks may leak metadata via packet timing.

Real-World Examples of SCAs in Parallel Systems

Cache-Based Attacks (Prime + Probe, Flush + Reload)

Attackers monitor cache usage to deduce cryptographic keys or sensitive data. For example, in AWS Lambda, researchers demonstrated cross-tenant cache attacks by analyzing memory access patterns of co-resident functions.

Timing Attacks in Parallel Algorithms

Parallel sorting or matrix multiplication algorithms may exhibit data-dependent execution times, allowing attackers to reconstruct inputs.

Network Covert Channels

In MPI-based HPC clusters, variations in communication latency can exfiltrate data between malicious nodes.

Detection and Mitigation Strategies

1. *Hardware isolation:* Technologies like Intel SGX and AMD SEV create secure enclaves to prevent cross-VM cache probing.
2. *Obfuscation techniques:* Adding noise to execution timings or memory access patterns disrupts SCA feasibility.
3. *Machine learning for anomaly detection:* Training models on hardware performance counters (HPCs) can flag suspicious cache behavior.

BACKGROUND AND FUNDAMENTALS OF SIDE-CHANNEL ATTACKS IN PARALLEL COMPUTING

Definition of Side-Channel Attacks: What Are Side-Channel Attacks?

Side-channel attacks (SCAs) are a class of security exploits that extract sensitive information by analyzing indirect signals such as power consumption, execution time, electromagnetic radiation, or cache access patterns, rather than targeting software vulnerabilities directly. Unlike traditional cyber-attacks (e.g., buffer overflows or SQL injection), SCAs exploit physical or microarchitectural behaviors of computing systems.

Relevance in Cloud and Parallel Computing

In cloud environments, where multiple users share physical hardware (CPUs, GPUs, memory), SCAs are particularly dangerous because:

- *Multi-tenancy:* Virtual machines (VMs) or containers from different users run on the same physical server, enabling attackers to spy on co-resident workloads.
- *Parallel processing:* Tasks running simultaneously on multi-core CPUs or GPUs may leak data through shared resources like caches or memory buses.
- *Distributed systems:* High-performance computing (HPC) clusters using MPI or RDMA can expose timing variations that reveal confidential data.

Historical Context: Major Side-Channel Vulnerabilities [3, 4]

Spectre and Meltdown (2018)

- *Spectre:* Exploits speculative execution in CPUs to trick applications into leaking data via cache timing. Affected nearly all modern processors.
- *Meltdown:* Breaks isolation between user and kernel memory, allowing unauthorized access to sensitive data.
- *Impact:* Cloud providers like AWS, Azure, and Google Cloud had to patch hypervisors and redesign VM isolation mechanisms.

Cache Bleed (2016)

- A cache-based attack targeting cryptographic operations by monitoring cache bank conflicts in Intel CPUs.
- Demonstrated that even constant-time algorithms (designed to resist timing attacks) could be vulnerable to microarchitectural leaks.

Foreshadow (L1TF, 2018)

- Exploited Intel's L1 cache speculative loading to steal data from SGX enclaves or other VMs.

Shared Cloud Environments and Parallel Computing: How They Enable SCAs

Virtualization and Multi-Tenancy

Cloud providers rely on virtualization (e.g., VMware, KVM) and containerization (e.g., Docker, Kubernetes) to maximize hardware utilization. However, this creates attack surfaces:

- *Example:* In AWS EC2, attackers can launch instances on the same physical host as a target VM and use Prime + Probe techniques to monitor cache usage, deducing cryptographic keys or private data.

Resource Sharing in Parallel Systems

Parallel computing architectures introduce additional risks due to shared hardware resources [5]:

1. *Shared CPU caches (L1/L2/L3)*
   a. *Problem:* Last-level caches (LLC) are shared across cores, allowing attackers to infer memory access patterns of victim processes.
   b. *Example:* In a multi-tenant GPU cloud, one user's CUDA kernel could probe another's cache lines to steal AI model weights.
2. *Memory deduplication*
   a. Cloud hypervisors optimize memory usage by merging identical pages (e.g., KSM in Linux).
   b. *Attack:* A malicious VM can detect whether a target VM shares memory pages, leaking information about running applications.

NETWORK-BASED SIDE CHANNELS

In MPI-based HPC clusters, attackers can infer communication patterns between nodes by analyzing packet timing, compromising sensitive simulations (e.g., weather forecasting, nuclear research).

Case Study: Cross-VM Attacks in Azure

1. Researchers demonstrated cache-based SCAs on Microsoft Azure by:
   a. Deploying a malicious VM on the same host as a victim VM.
   b. Using Flush + Reload to monitor cache accesses during AES encryption.
   c. Recovering the full encryption key via statistical analysis.
2. *Mitigation:* Azure now uses core scheduling and cache partitioning to isolate workloads.

Why Parallel Computing Magnifies the Risk

- *High-throughput workloads:* AI training, scientific simulations, and big data processing generate predictable memory/CPU patterns, making SCAs easier.

- *Distributed coordination:* Frameworks like Apache Spark or TensorFlow distribute tasks across nodes, creating timing side channels.

TYPES OF SIDE-CHANNEL ATTACKS IN PARALLEL SYSTEMS

Side-channel attacks (SCAs) in parallel computing environments leverage various hardware and software characteristics to extract sensitive information. These attacks are particularly dangerous in shared cloud infrastructures where multiple users compete for the same physical resources. Below we examine four major categories of SCAs with concrete examples demonstrating their real-world impact [6–9].

Cache-Based Attacks

Cache-based attacks exploit CPU cache sharing between processes to steal information. Modern processors use multi-level caches (L1/L2 private per core, L3 shared across cores), creating opportunities for cross-process spying.

Prime + Probe Attack

1. *Mechanism*
   a. Attacker fills a cache set with their own data ("Prime").
   b. Waits for victim to potentially access same cache lines.
   c. Measures time to re-access their data ("Probe"); slower access indicates victim usage.
2. *Example*
   a. In AWS Lambda, researchers extracted RSA keys by monitoring cache contention between co-resident functions.
   b. Required no special permissions, just ability to run on same physical CPU.

Flush + Reload Attack

1. *Mechanism*
   a. Attacker flushes shared memory region from cache.
   b. Waits then measures reload time; faster reload indicates victim accessed the data.
2. *Example*
   a. Used to break kernel address space layout randomization (KASLR) in cloud VMs.
   b. Allowed mapping of protected memory regions before Meltdown/Spectre exploits.

Cloud Impact

1. Google Cloud had to redesign their serverless infrastructure to prevent cross-tenant cache attacks.
2. Azure implemented cache partitioning for its confidential computing offerings.

Timing Attacks

Timing attacks analyze variations in algorithm execution time to deduce secret data. Parallel systems amplify these risks due to complex scheduling behaviors.

Branch Prediction Attacks

1. *Mechanism*
   a. Measure timing differences in speculative execution paths.
   b. Modern CPUs predict branches, creating data-dependent timing variations.
2. *Example*
   a. Spectre variants exploited this to read memory across security boundaries.
   b. Cloud providers had to disable hyperthreading for high-security workloads.

Parallel Algorithm Vulnerabilities

1. *Matrix multiplication*
   a. Execution time varies based on input data patterns.
   b. Researchers reconstructed neural network weights by timing GPU operations.

2. *Database operations*
    a. Query time leaks information about data distribution.
    b. Amazon Aurora had to implement constant-time query processing.

Mitigation Challenges

1. True constant-time algorithms are extremely difficult in parallel environments.
2. Cloud providers now use noise injection to obscure timing channels.

Power and Electromagnetic Analysis

These hardware-level attacks measure physical emissions from computing equipment to extract secrets.

Power Analysis Attacks

1. *Mechanism*
    a. Monitor power consumption fluctuations during computation.
    b. Cryptographic operations create distinct power signatures.
2. *Cloud example:* Researchers recovered RSA keys from adjacent VMs in IBM Cloud by:
    a. Deploying malicious VM on same physical host.
    b. Using Intel RAPL interface to sample power usage.
    c. Applying differential power analysis techniques.

Electromagnetic (EM) Attacks

1. *Data center risks*
    a. Server racks emit compromising EM signals.
    b. Requires physical proximity but possible in colocation facilities.
2. *Case study:* French government reported attacks against cloud providers where:
    a. Attackers rented space in target data center.
    b. Used specialized equipment to capture EM leaks from SSL handshakes.

Defense Approaches

1. Faraday cage shielding for high-security systems.
2. Power line filtering and signal masking technologies.

Network-Based Side Channels

Distributed parallel computing introduces network timing channels that can leak sensitive information.

MPI Covert Channels

1. *Mechanism*
    a. Manipulate message timing/scheduling in MPI clusters.
    b. Encode data in communication patterns.
2. *HPC example:* Attackers reconstructed confidential simulation parameters in weather forecasting clusters by:
    a. Observing message timing between compute nodes.
    b. Analyzing synchronization patterns in Allreduce operations.

RDMA Vulnerabilities [10]

1. *Problem:* RDMA's kernel bypass enables ultra-low latency but:
    a. Memory access patterns visible to other nodes.
    b. Timing variations leak information.
2. *Cloud impact:* Azure HPC had to modify its RDMA stack after researchers demonstrated:
    a. Cross-tenant memory access pattern reconstruction.
    b. Bandwidth monitoring attacks in InfiniBand networks.

Mitigation Strategies

1. Network traffic padding and randomization.
2. QoS controls to prevent timing measurements.
3. Secure RDMA memory registration protocols.

Emerging Threats and Future Directions

New attack vectors continue to emerge as parallel computing evolves:

1. *GPU side channels*
   a. Memory access patterns in CUDA kernels leak AI model information.
   b. Demonstrated against TensorFlow workloads in cloud GPUs.
2. *Quantum computing risks*
   a. Qubit crosstalk may create new side channels.
   b. Already observed in early quantum cloud services.
3. *Persistent memory attacks*
   a. Intel Optane DC PMEM shows cache-like side effects.
   b. New attack surfaces in memory-centric computing.

Defensive Trends

1. Hardware-enforced isolation (Intel TDX, AMD SEV-SNP).
2. Machine learning for anomaly detection in HPC clusters.
3. Formal verification of parallel algorithms.

This comprehensive analysis demonstrates that side-channel threats permeate all layers of parallel systems, requiring continuous security innovation across hardware, software, and operational practices. Cloud providers and HPC facilities must adopt defense-in-depth strategies combining technical controls with rigorous security monitoring.

DETECTION AND ANALYSIS TECHNIQUES FOR SIDE-CHANNEL ATTACKS

As side-channel attacks grow more sophisticated, researchers have developed advanced detection methods combining machine learning, hardware monitoring, and formal verification. These techniques are particularly crucial for parallel computing environments where shared resources create complex attack surfaces.

Anomaly Detection Using Machine Learning

Modern ML approaches have proven effective at identifying subtle side-channel attack patterns [11–13]:

1. *Approaches*
   a. *Supervised learning*
      i. Trains classifiers on labeled datasets of normal vs. attack behaviors.
      ii. *Example:* Random Forest models detecting cache access anomalies with 92% accuracy in AWS Lambda.
   b. *Unsupervised anomaly detection*
      i. Uses clustering (e.g., Isolation Forest) to flag deviations from normal patterns.
      ii. Applied successfully to detect novel Spectre variants in Google Cloud.
   c. *Deep learning*
      i. LSTMs analyze temporal patterns in hardware performance counters.
      ii. Achieved 96% detection rate for Flush + Reload attacks in Azure VMs.
2. *Implementation challenges*
   a. Requires extensive training data from production systems.
   b. Must balance detection rates with false positives (critical for cloud providers).
   c. NVIDIA developed specialized ML models for GPU workload monitoring.

Hardware Performance Counters (HPCs)

HPCs provide low-level system metrics that reveal attack signatures:

1. *Key counters for SCA detection [12]*
    a. *Cache monitoring*
        i. LLC_REFERENCES, LLC_MISSES expose cache contention.
        ii. Detected 89% of Prime + Probe attacks in research studies.
    b. *Branch prediction*
        i. BRANCH_MISPREDICTIONS helps identify Spectre-like attacks.
        ii. Used in Intel's Counterfeit vulnerability detection system.
    c. *Memory access*
        i. MEM_LOAD_RETIRED tracking reveals memory access patterns.
        ii. Amazon Guard Duty employs these for EC2 instance monitoring.
2. *Deployment considerations*
    a. Overhead typically <3% for continuous monitoring.
    b. Cloud providers increasingly expose HPCs securely to tenants.
    c. ARM's PMU and Intel's PEBS provide architectural support.

Formal Verification Methods

Mathematical approaches to prove algorithm resistance:

1. *Techniques*
    a. *Information flow analysis*
        i. Tracks how data propagates through systems.
        ii. Verified OpenSSL's constant-time RSA implementation.
    b. *Model checking*
        i. Exhaustively explores possible execution paths.
        ii. Used to verify SGX enclave designs at Intel.
    c. *Theorem proving*
        i. Mathematical proofs of security properties.
        ii. Applied to seL4 microkernel for side-channel resistance.
2. *Real-world applications*
    a. AWS uses formal methods to verify Nitro hypervisor security.
    b. Google's Project Zero employs verification for patch validation.
    c. Microsoft's Everest framework verifies cryptographic implementations.
3. *Emerging directions*
    a. Combining ML and formal methods for hybrid analysis.
    b. Quantum-resistant algorithm verification.
    c. Automated proof generation for parallel algorithms.

Cloud providers increasingly combine these methods; for example, Azure Stack HCI uses HPCs for real-time detection backed by ML analysis, with formal verification for critical components. This layered approach is becoming standard for securing parallel computing infrastructures against evolving side-channel threats. Future work focuses on reducing false positives in ML models and making formal methods more scalable for complex parallel systems.

MITIGATION STRATEGIES AGAINST SIDE-CHANNEL ATTACKS IN PARALLEL COMPUTING

As side-channel attacks become more sophisticated, researchers and cloud providers have developed multi-layered defense strategies. These mitigation approaches range from hardware-level isolation to software-based obfuscation techniques, each addressing different attack vectors in parallel computing environments.

Isolation Techniques

Hardware-enforced isolation provides the strongest defense against cross-tenant attacks:

1. *Secure enclaves*
   a. *Intel SGX:* Creates encrypted memory regions (enclaves) that even the OS/hypervisor cannot access. Used in Azure Confidential Computing for protecting sensitive data processing.
   b. *AMD SEV-SNP:* Encrypts VM memory with unique keys and prevents memory integrity attacks. Google Cloud's Confidential VMs leverage this for secure multi-tenant workloads.
2. *Cache partitioning*
   a. Intel CAT (Cache Allocation Technology) and AMD's equivalent allow dedicating cache portions to specific VMs/processes.
   b. Cloudflare uses cache partitioning to prevent cross-site attacks in their edge servers.
3. *Implementation challenges*
   a. Performance overhead (5–30% depending on workload).
   b. Requires hardware support not available in all systems.

Noise Injection and Obfuscation

These software techniques make side-channel signals unreliable:

1. *Timing obfuscation*
   a. Adding random delays to memory accesses and computations.
   b. Google's Project Zero implemented this for Chrome's cryptographic operations.
2. *Memory access randomization*
   a. Frequently shuffling memory locations of sensitive data.
   b. Used in Linux kernel defenses against Meltdown-type attacks.
3. *Effectiveness*
   a. Reduces attack accuracy by 60–80% in studies.
   b. Minimal performance impact (typically <5%).

Architectural Solutions

Hardware modifications provide fundamental protections [13]:

1. *Constant-time execution*
   a. Eliminates data-dependent timing variations.
   b. ARM's MTE (Memory Tagging Extension) helps achieve this.
2. *Speculation control*
   a. Intel's eIBRS (Enhanced Indirect Branch Restricted Speculation).
   b. AMD's SSB (Speculative Store Bypass) mitigation.
3. *Emerging architectures*
   a. RISC-V with built-in side-channel resistance features.
   b. IBM's secure processor designs for cloud environments.

Policy and Access Control

Administrative measures complement technical defenses:

1. *Scheduling policies*
   a. Core dedication for sensitive workloads.
   b. AWS's Nitro System enforces strict VM-to-core pinning.
2. *Access control models*
   a. Role-based memory access permissions.
   b. Kubernetes Pod Security Policies for container isolation.
3. *Cloud-specific implementations*
   a. Azure's Confidential Computing attestation.
   b. Google's BeyondProd zero-trust architecture.

Modern cloud providers typically employ all four strategies in combination. For example, Azure's DCsv3-series VMs use AMD SEV-SNP (isolation), constant-time crypto (architectural), memory randomization (obfuscation), and strict access policies. This defense-in-depth approach is crucial as attackers increasingly combine multiple side-channel techniques.

CHALLENGES AND FUTURE DIRECTIONS IN SIDE-CHANNEL ATTACK MITIGATION

Balancing Security and Performance Overhead

One of the most pressing challenges in defending against side-channel attacks is maintaining system performance while implementing robust security measures. Current mitigation techniques often impose significant computational overhead [14]:

- Secure enclaves like Intel SGX can reduce performance by 20–40% for certain workloads.
- Cache partitioning strategies may decrease overall throughput by 15–25%.
- Constant-time cryptographic algorithms typically run 2–3x slower than their vulnerable counterparts.

The trade-off is particularly acute in high-performance computing environments where every cycle counts. Researchers are exploring several approaches to optimize this balance:

- *Selective hardening:* Applying protections only to security-critical code paths.
- *Hardware-accelerated security:* New CPU instructions for faster encryption and isolation.
- *Adaptive defenses:* Systems that dynamically adjust protection levels based on threat detection.

Quantum-Resistant Parallel Computing

The advent of quantum computing introduces new side-channel vulnerabilities while also demanding novel protection mechanisms [13, 14]:

1. *Quantum side-channel threats*
   a. Qubit crosstalk in multi-core quantum processors.
   b. Power analysis attacks on quantum control electronics.
2. *Post-quantum cryptography challenges*
   a. Larger key sizes (2–4x) in lattice-based algorithms increase memory access patterns.
   b. Complex mathematical operations create new timing attack surfaces.

Leading cloud providers are already testing hybrid systems that combine classical and quantum-resistant algorithms, with Google Cloud implementing preliminary support for CRYSTALS-Kyber key encapsulation in its HPC offerings.

Standardized Benchmarks for SCA Resilience

The lack of comprehensive evaluation standards hampers effective security comparisons:

1. *Current limitations [15]*
   a. No agreed-upon metrics for quantifying side-channel resistance.
   b. Vendor-specific testing methodologies that cannot be directly compared.
   c. Inadequate simulation of real-world attack scenarios.
2. *Emerging solutions*
   a. NIST's on-going development of a side-channel assessment framework.
   b. Academic efforts like the SCA Evaluation Board (SCAEB) project.
   c. Cloud Security Alliance's working group on microarchitectural vulnerabilities.

Key Benchmarking Needs

1. Performance/security trade-off quantification.
2. Attack surface coverage metrics.
3. Cross-platform evaluation methodologies.

CONCLUSION

Side-channel attacks (SCAs) represent a persistent and evolving threat to parallel computing systems, particularly in shared cloud environments where multi-tenancy and resource sharing create inherent vulnerabilities. This review has examined the spectrum of SCAs, from cache-based and timing attacks to power analysis and network-based exploits, demonstrating their severe implications for data confidentiality and system integrity in high-performance computing (HPC), AI workloads, and distributed systems. Notable examples like Spectre, Meltdown, and cross-VM cache attacks underscore the real-world risks, forcing cloud providers and hardware manufacturers to continuously adapt their defenses. The discussion of mitigation strategies highlighted a layered approach to security, combining hardware-enforced isolation (e.g., Intel SGX, AMD SEV), software-based obfuscation (e.g., noise injection), architectural innovations (e.g., constant-time execution), and stringent access controls. While these measures significantly reduce attack surfaces, challenges remain in balancing security with performance overhead, especially for latency-sensitive applications. Furthermore, the emergence of quantum computing introduces new attack vectors, demanding proactive research into post-quantum cryptography and quantum-resistant architectures. Looking ahead, three critical areas require attention: (1) optimizing security-performance trade-offs through hardware-software co-design, (2) standardizing benchmarks to evaluate SCA resilience across platforms, and (3) fostering collaboration between industry and academia to address novel threats. The development of robust, scalable defenses will be essential as parallel computing expands into exascale systems, edge environments, and quantum-HPC hybrids. Ultimately, securing parallel systems against SCAs is not a one-time effort but an on-going arms race. By integrating cutting-edge research, cross-disciplinary insights, and proactive policy frameworks, the computing community can mitigate these risks and uphold the trustworthiness of next-generation infrastructures. Future advancements must prioritize both theoretical rigor and practical deployability to ensure that performance gains in parallel computing do not come at the cost of compromised security.

REFERENCES

1. Abadi M, Blanchet B, Fournet C. Just fast keying in the pi calculus. ACM Trans Inf Syst Secur. 2007 Jul 1; 10(3): 9-es.
2. Bernstein DJ. Cache-timing attacks on AES. Chicago, IL: The University of Illinois at Chicago; 2005. p.60607–7045. http://wistp2007.wistp.org/fileadmin/damiensauveron/Cours/M2/certification/Attacks/TimingAttack/cachetiming-20050414.pdf
3. Canella C, Van Bulck J, Schwarz M, Lipp M, Von Berg B, Ortner P, Piessens F, Evtyushkin D, Gruss D. A systematic evaluation of transient execution attacks and defenses. In 28th USENIX Security Symposium (USENIX Security 19). 2019; 249–266.
4. Ge Q, Yarom Y, Cock D, Heiser G. A survey of microarchitectural timing attacks and countermeasures on contemporary hardware. J Cryptogr Eng. 2018 Apr; 8: 1–27.
5. Gruss D, Maurice C, Mangard S. Rowhammer. js: A remote software-induced fault attack in javascript. In Detection of Intrusions and Malware, and Vulnerability Assessment: 13th International Conference, DIMVA 2016, San Sebastián, Spain, July 7–8, 2016, Proceedings 13. Cham: Springer International Publishing; 2016; 300–321.
6. Kocher P, Horn J, Fogh A, Genkin D, Gruss D, Haas W, Hamburg M, Lipp M, Mangard S, Prescher T, Schwarz M. Spectre attacks: Exploiting speculative execution. Commun ACM. 2020 Jun 18; 63(7): 93–101.
7. Lipp M, Schwarz M, Gruss D, Prescher T, Haas W, Horn J, Mangard S, Kocher P, Genkin D, Yarom Y, Hamburg M. Meltdown: Reading kernel memory from user space. Commun ACM. 2020 May 21; 63(6): 46–56.
8. Osvik DA, Shamir A, Tromer E. Cache attacks and countermeasures: the case of AES. In Topics in Cryptology–CT-RSA 2006: The Cryptographers' Track at the RSA Conference 2006, San Jose, CA, USA, February 13-17, 2005. Proceedings. Berlin Heidelberg: Springer; 2006; 1–20.
9. Schwarz M, Lipp M, Moghimi D, Van Bulck J, Stecklina J, Prescher T, Gruss D. ZombieLoad: Cross-privilege-boundary data sampling. In Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security. 2019 Nov 6; 753–768.

10. Van Bulck J, Minkin M, Weisse O, Genkin D, Kasikci B, Piessens F, Silberstein M, Wenisch TF, Yarom Y, Strackx R. Foreshadow: Extracting the keys to the intel {SGX} kingdom with transient {Out-of-Order} execution. In 27th USENIX Security Symposium (USENIX Security 18). 2018; 991–1008.
11. Wang Z, Lee RB. Covert and side channels due to processor architecture. In 2006 IEEE 22nd Annual Computer Security Applications Conference (ACSAC'06). 2006 Dec 11; 473–482.
12. Yarom Y, Falkner K. {FLUSH+ RELOAD}: A high resolution, low noise, l3 cache {Side-Channel} attack. In 23rd USENIX security symposium (USENIX security 14). 2014; 719–732.
13. Zhang Y, Juels A, Reiter MK, Ristenpart T. Cross-tenant side-channel attacks in PaaS clouds. In Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security. 2014 Nov 3; 990–1003.
14. Intel. (2018). Analysis of Speculative Execution Side Channels. [Online]. Available from: https://www.intel.com/content/dam/www/public/us/en/documents/white-papers/analysis-of-speculative-execution-side-channels-white-paper.pdf
15. AMD. (2025). AMD Secure Encrypted Virtualization (SEV). [Online]. Available from: https://www.amd.com/en/developer/sev.html

