Annals of Emerging Technologies in Computing (AETiC)
Vol. 10, No. 2, 2026

Research Article

Enhanced Database Security using Hybrid GA-PSO for Parallel Elliptic Curve Cryptographic Scheduling with Offline Optimisation

Safaa Salam Hatem* **and Fahad Naim Nife**

Al-Muthanna University, Muthanna, Iraq
safaa.salam@mu.edu.iq; fahad.naim@mu.edu.iq
*Correspondence: safaa.salam@mu.edu.iq

Received: 21 July 2025; Accepted: 31 March 2026; Published: 1 April 2026

Abstract: Modern database systems increasingly employ cryptographic primitives; however, the significant computational overhead of encryption and decryption represents a major performance bottleneck in high-throughput environments. In particular, Elliptic Curve Cryptography (ECC) is of interest because it provides strong security guarantees with compact key sizes and relatively efficient arithmetic, thus making ECC suitable for resource-constrained platforms and latency-sensitive services. Realistic database workloads require the concurrent execution of many ECC scalar multiplications, such as batch encryption pipelines, parallel client authentication, and transaction-intensive workloads, hence inducing a non-trivial scheduling problem on modern multi-core and heterogeneous architectures. This work introduces a hybrid Genetic Algorithm-Particle Swarm Optimisation (GA-PSO) framework that acts as an offline configuration mechanism for optimizing parallel ECC operation execution on heterogeneous resources, which runs in a pre-computation phase: the optimisation routine is executed only once during system initialization, or when the database is deployed on new hardware, and the scheduling parameters are cached and reused for the system's operational lifetime. In this way, the optimisation cost is amortized over millions of subsequent cryptographic operations, reconciling the apparent trade-off between sophisticated scheduling strategies and the stringent performance constraints of real-time database workloads; Moreover, the new scheduler has a four-dimensional decision space: ECC window size, core assignment, run ordering, and memory placement, and it utilizes a five-dimensional score function which considers runtime, memory footprint, load balancing, energy consumption, and schedule predictability. Experimental evaluation on the widely deployed secp256r1 curve shows that the Offline GA-PSO scheduler improves the execution time by 4.2% compared to the dynamic Work-Stealing baseline, by 9.8% compared to Greedy SJF and by 24.5% compared to Round-Robin, and given an offline optimisation cost of 0.85 s and a per-batch saving of 9.5 ms relative to the strongest runtime baseline, this overhead is amortised after approximately 4,500 scalar multiplications. Overall, our results indicate that offline meta-heuristic scheduling is a viable and efficient building block for smart, crypto-aware resource management in secure high-performance database systems, and moreover, the experimental evaluation shows that on the target multi-core platform the proposed offline hybrid GA-PSO scheduler converges most frequently to an ECC window size of 8, as this choice offers the best trade-off between precomputation overhead and scalar multiplication speed.

Keywords: ***Cryptographic scheduling; Database security; Elliptic Curve Cryptography (ECC); Genetic Algorithm (GA); Multi-core optimisation; Offline optimisation; Particle Swarm Optimisation (PSO); Resource management***

1. Introduction

1.1. Database Security in the Modern Computing Landscape

The tremendous proliferation of data-intensive applications has established database systems as a critical infrastructure in many sectors, and financial institutions are currently processing millions of transactions on a daily basis, while health-care providers are maintaining large repositories of protected

Safaa Salam Hatem and Fahad Naim Nife, "Enhanced Database Security using Hybrid GA-PSO for Parallel Elliptic Curve Cryptographic Scheduling with Offline Optimisation", *Annals of Emerging Technologies in Computing (AETiC)*, Print ISSN: 2516-0281, Online ISSN: 2516-029X, 1 April 2026, Vol. 10, No. 2, pp. 35-52, Published by International Association for Educators and Researchers (IAER), DOI: 10.33166/AETiC.2026.02.002, Available: http://aetic.theiaer.org/archive/v10/v10n2/p2.html.

health information, and cloud service providers are managing access to highly sensitive business data in large-scale distributed environments. The increased concentration of valuable digital assets has transformed databases into a prime target for sophisticated cyber attacks, including external breaches, advanced persistent threats, and malicious insider attacks [1-2], because regulatory frameworks such as the General Data Protection Regulation (GDPR) and the Health Insurance Portability and Accountability Act (HIPAA) place stringent obligations on data controllers and processors to employ robust encryption mechanisms to ensure the confidentiality and integrity of stored and transmitted data [3].

Modern database security architectures typically require making a fundamental trade-off between strong cryptographic protection and high-throughput operation. In a typical enterprise deployment, a database server will support thousands of concurrent client connections, and each of these connections may require cryptographic processing for TLS handshakes, column-level encryption, and authenticated query execution. If these cryptographic operations are performed in a purely serialized manner, they will quickly become a major performance bottleneck and compromise system scalability, because naive parallelization strategies that assign cryptographic tasks to available cores ignore the complex resource constraints, timing properties, and side-channel-related security constraints of cryptographic processing, and can result in both poor performance and subtle security vulnerabilities [4].

1.2. Efficiency and Ongoing Relevance of Elliptic Curve Cryptography

Elliptic Curve Cryptography (ECC) was first introduced in the 1980s as a competitive alternative to RSA, offering the same security guarantees with much smaller key sizes. For example, a 256-bit ECC key is widely accepted as providing a security level equivalent to that of a 3072-bit RSA key, which results in faster cryptographic computations, lower communication overhead for shorter key and signature encodings, and lower memory footprints in both software and hardware implementations [5-6], and these features make ECC particularly appealing for resource-constrained platforms, including mobile devices, IoT sensors, smart cards and deeply embedded systems, where processing power, energy budgets and memory are all limited. Although the cryptographic community is in the process of a gradual transition to post-quantum cryptography, ECC remains highly relevant for several reasons: first, current NIST recommendations explicitly support hybrid cryptographic schemes that combine classical public-key algorithms like ECC with post-quantum constructions, to offer defence-in-depth during the transition period [7-8]; second, the vast installed base of ECC-based infrastructures, in protocols, libraries, hardware security modules and application stacks, implies that ECC will continue to be widely used for many years to come, even as post-quantum algorithms are gradually adopted; and third, many resource-constrained IoT and embedded devices lack the processing power and/or energy required to efficiently implement lattice-based post-quantum schemes, at least in their current standardized forms, so optimised ECC implementations are necessary to enable practical and secure communication in such scenarios [9-10].

1.3. The Parallel Scalar Multiplication Scheduling Challenge

Scalar multiplication operations for ECC are rarely isolated in practical databases, and they occur as sets or batches as a result of user authentication, secure query evaluation, transactions, and group cryptographic operations. As such, when they execute concurrently on multiple cores, they no longer represent a simple problem of task-core mapping, because they involve a complicated trade-off between scheduling and configuration choices, where performance, memory footprint, and consistency are all interdependent. Parallel ECC scheduling is modeled by a four-dimensional decision vector: (i) number of cores to be assigned to each scalar multiplication; (ii) which operations should be performed by each core; (iii) where to store precomputation tables and shared ECC data; and (iv) size of the window to be used for trading-off compute costs against table sizes, and we measure the quality of such decisions through a five-objective fitness function. The function captures execution time, memory hierarchy behaviour, load balance, energy consumption, and predictability of the schedule [11-12], because the objectives are highly correlated. A high-quality parallel ECC scheduler must simultaneously account for performance, latency, core utilisation, cache behaviour, and predictability, therefore it cannot afford to treat ECC as a generic parallel

task. This motivates our offline optimisation strategy, described in the next section, thus a comprehensive approach is required.

1.4. The Offline Optimisation Paradigm

The problem of meta-heuristic optimisation in cryptographic workloads is, in a nutshell, a paradox: it is, prima facie, highly unattractive to use a large, slow optimisation process to speed up a set of small, fast cryptographic algorithms. One of the key questions to be answered in this chapter is, therefore, whether the overhead of the GA-PSO is sufficient to cancel out the speedup, and it is crucial to answer this question when evaluating the practicality of the proposed scheduling framework. To resolve the paradox, it is necessary to consider when, where and how the optimisation is to be executed, and, in particular, whether the one-off cost of the GA-PSO run can be amortized over the entire operational lifetime of the database system. If the optimisation can be limited to an offline pre-computation phase and its results reused in millions of subsequent ECC operations, then the "sledgehammer" is a one-off cost that provides performance improvements at runtime over a long period, reconciling advanced meta-heuristic scheduling with the strict latency and throughput requirements of modern cryptographic database services. This paradox is resolved by the proposed framework through an intentional architectural choice: the GA-PSO optimiser is designed as an offline pre-computation module (as opposed to a real-time online scheduler), and this distinction is key to understanding why the approach is computationally valid. The offline optimisation approach works as follows in the proposed framework: the GA-PSO algorithm is executed only once, at system initialization time, at hardware deployment time, or when there is a large and persistent change in workload characteristics, and it is not executed for every cryptographic operation, or even every batch. Once the optimiser converges, the resulting configuration, which includes core assignments, execution orderings, memory placement strategies, and ECC window size selections, is cached and reused for the entire operational phase of the system, until a large hardware or workload change occurs. The cost of a single optimisation pass (on the order of 0.85 s in our experiments with 50 operations) is then amortized over the millions of cryptographic operations executed in the following days, weeks or months, and its perceived overhead becomes negligible in practice. Additionally, when the framework is intended for deployment in heterogeneous environments, such as IoT networks, FPGA-accelerated platforms or multi-core servers with different microarchitectural characteristics, the optimiser produces a customized configuration for each hardware profile prior to production deployment, so that every platform starts its operating life with a scheduling policy that matches its architectural characteristics, thereby leveraging platform-specific performance features without paying online tuning costs. To put the benefits of the offline GA-PSO into perspective for real-world deployment, let us consider a DB server executing approximately 10,000 ECC operations per hour, in 200 batches of 50 operations each. The GA-PSO process is called once at deployment time, with a one-off offline optimization cost of 0.85 s, and the produced configuration achieves an execution time saving of 9.5 ms per batch with respect to the best runtime baseline from Table 2, the dynamic Work-Stealing scheduler (224.8 ms vs 215.3 ms). At 200 batches per hour, this equates to a saving of about 1.9 s per hour, so the one-off optimization cost is amortized after approximately 27 minutes of normal operation. The cumulative saving over 24 h is about 45.6 s, and over a 30 day month about 1,368 s, therefore, these numbers show that the offline GA-PSO does not only solve the "Sledgehammer Paradox," but also offers a clear deployment-time value proposition for common cryptographic workloads in DB environments.

1.5. Decision Variables and Multi-Objective Fitness Design

The proposed framework separates cleanly between decision variables and evaluation objectives, and every candidate configuration is represented as a four-dimensional vector S= (σ, π, μ, w), where σ encodes the core allocation, π the execution ordering, μ the memory placement and w the ECC window size. The four variables form the search space explored by the hybrid GA-PSO algorithm because they are the fundamental components of the configuration. Note that energy is not the fifth decision variable; rather, every candidate vector S is evaluated with a five-objective fitness function that evaluates jointly (i) execution time, (ii) memory-hierarchy behaviour, (iii) load balance, (iv) energy consumption and (v) scheduling-level

predictability, therefore the framework does not optimize over a five-dimensional solution vector, but searches over four decision variables under a five-objective evaluation model. For 50 ECC operations mapped onto 8 cores, the size of the search space is determined by the combinatorial possibilities of σ, π, μ and w, and the fitness function determines the desirability of each candidate configuration with respect to the five objectives listed above. This combinatorial complexity is the very reason why a meta-heuristic approach like GA-PSO is needed instead of simple handcrafted rules or static lookup tables.

1.6. Research Objectives and Contributions

The studies discussed above identify a disconnect between the efficiency of cryptographic execution and secure resource management. This work addresses the above disconnect through the following five contributions: first, we formulate parallel ECC execution as an offline optimisation problem, thereby demonstrating that the one-off GA-PSO search overhead can be amortized across a large number of subsequent cryptographic operations. Second, we define a multi-dimensional decision vector S = (σ, π, μ, w) that jointly models core allocation, execution ordering, memory placement, and window-size selection under an energy-aware objective. Third, we present a hybrid GA-PSO procedure and corresponding deployment algorithm that converts the optimised vector into a reusable static scheduling policy for recurring workloads. Fourth, we present an experimental evaluation of the proposed approach against multiple baseline schedulers, including a dynamic Work-Stealing baseline, while providing convergence analysis, window-size sensitivity analysis, and workload-variation experiments. Fifth, we introduce a security-aware scheduling perspective where timing dispersion and conflict-prone memory behaviour are incorporated as scheduling-level design considerations, but note that the present work does not claim a formal proof of constant-time cryptographic execution, nor does it implement a full physical leakage evaluation such as TVLA.

2. Related Work

Parallel cryptographic execution is related to research from multiple partially overlapping fields: algorithm-level ECC optimisation, hardware acceleration, meta-heuristic resource management, and side-channel analysis. The problem that this paper solves falls precisely at the intersection of these fields: database-oriented ECC workloads require parallelism to meet latency and throughput constraints, yet straightforward scheduling increases contention and threatens predictability. This section therefore discusses the relevant literature in a purposeful order: it first discusses the justification of parallelism, then presents performance-oriented meta-heuristic scheduling, and finally exposes the dangers of crypto-oblivious schedulers, which therefore motivates the need for an offline, crypto-aware optimisation framework that explicitly considers security-relevant execution properties when scheduling.

2.1. Parallel Processing Architectures for ECC

Although ECC is attractive for its high security for relatively small key sizes, contemporary databases and networked systems invoke ECC repeatedly, not sporadically, because parallel client authentication, batched key exchange, encrypted transaction processing, and secure background services can all lead to bursts of scalar multiplications in a short time window. In such cases, the ability to exploit parallel execution on multi-core CPUs and heterogeneous platforms is an operational necessity, not a performance nicety, and prior hardware-oriented research has shown that substantial latency reductions are possible if special-purpose accelerators such as FPGAs or ASICs are available [17-19]. However, typical software-centric deployments, especially database servers, have a fundamentally different primary problem: it is not just how fast a single ECC operation can be executed, but how many concurrent ECC operations can be scheduled efficiently and predictably on shared computational resources without degrading overall system behaviour.

2.2. Computational Bottlenecks in ECC Scalar Multiplication

The primary computational expense in ECC is still scalar multiplication, which typically accounts for most of the running time in high-level ECC protocols [13-15]. Recent work continues to enhance the

performance of one operation at a time through arithmetic optimisations and precomputation techniques, for example, M-ary precomputation and related techniques can greatly reduce the latency of a single scalar multiplication [16]. However, these optimisations focus on the algorithmic efficiency of one operation at a time and do not address the system-level issues that arise in database workloads, where many ECC operations are concurrently competing for cores, cache, and memory bandwidth. Thus, existing literature still does not have a framework for concurrency, execution order, cache behaviour, and deployment-level constraints for large batches of ECC tasks because it is exactly this lack that moves the problem from purely arithmetic optimisation to a more general scheduling and resource management problem.

2.3. Meta-Heuristics for Cryptographic Optimisation

Hybrid meta-heuristics, such as GA-PSO, have shown good promise in searching large, constrained solution spaces, particularly in engineering design and distributed resource allocation problems [20-22], because their major advantage is that they can take advantage of the strengths of broad evolutionary exploration and more focused swarm-based exploitation, which enables an efficient search of decision spaces that are too large or too irregular to be searched exhaustively or purely analytically. However, the objective functions considered in these studies are crypto-oblivious, i.e. they optimize latency, throughput or energy in the presence of generic system constraints, without explicitly encoding execution properties that are critical for cryptographic workloads, and therefore a crypto-aware scheduler needs to incorporate the additional security-relevant considerations, including timing regularity, resilience to contention-induced variability, safe placement of precomputation tables in memory, potential core isolation for key-dependent tasks and safe runtime adaptations during the cryptographic sections. The lack of security-relevant objectives in the prior GA-PSO scheduling studies creates a significant gap between general-purpose optimisation and the requirements of secure cryptographic execution.

2.4. Side-Channel Risks in Dynamic Cryptographic Algorithms

Cryptographic performance cannot be evaluated independently of leakage risk because timing attacks, power analysis and cache-based attacks all rely on the same behavioural variations of the execution environment [23-25], and while dynamic runtime techniques may indeed increase utilisation, they can also add additional scheduler-driven variability that degrades predictability and amplifies existing leakage channels, especially in the case of ECC schemes that rely on precomputation tables or other memory-sensitive optimisations. A scheduler for cryptographic workloads should not, therefore, be performance-driven alone, but should avoid pathological contention patterns and unnecessary runtime adaptations that can be observed or probed by an attacker, because this is not to say that scheduling alone is responsible for side-channel resistance, but it is a major enabler that can either reinforce or undermine the overall predictability and leakage profile of the deployed system.

2.5. Offline Optimisation for Heterogeneous Cryptographic Workloads

Secure systems are becoming increasingly heterogeneous both in terms of hardware and cryptographic composition: modern deployments will combine general-purpose CPUs with special-purpose accelerators and will, during the transition period, deploy both classical primitives (e.g. ECC) and post-quantum primitives [7-8], [27-28], and in such settings, the one-time offline optimisation paradigm is particularly attractive because it can learn a hardware-specific configuration prior to deployment and then use this configuration for a repeated workload. Our approach differs from static hardware design and fully dynamic runtime scheduling because it preserves the ability to explore a large configuration space, yet provides simple and predictable runtime scheduling behaviour; therefore, for database-oriented cryptographic services, the offline paradigm offers a pragmatic compromise between the rigidity of hand-tuned configurations and the crypto-agnostic nature of highly adaptive schedulers.

2.6. Comparison of Key Recent Works

Table 1 lists several representative publications in ECC acceleration, meta-heuristic optimisation and side-channel-aware design, and in general, these works focus on either arithmetic speed, hardware

acceleration or generic scheduling efficiency individually. However, the overlapping of offline scheduling, memory-aware execution and explicitly security-aware objective design has not been well addressed, which motivates the proposed integrated methodology.

Table 1. Critical Comparison of Recent ECC and Cryptographic Optimisation Research

Reference	Core Contribution	Target Problem	Methodology	Critical Limitations	Relevance to Scheduling
Wu *et al.* [16]	M-ary precomputation, 59% time reduction	Single ECSM optimisation	Mathematical analysis	Static parameter; single-operation focus	No multi-operation scheduling
Cui *et al.* [17]	FPGA accelerator (0.56ms)	Hardware acceleration	Parallel Montgomery	Static design; no adaptability	Requires intelligent offloading
Daftardar [18]	ZKP ASIC (400x speedup)	MSM acceleration	HLS ASIC design	Protocol-specific; no scheduling	Needs task distribution
Nwogbaga [22]	GA-PSO for IoT offloading	General task scheduling	Simulation (iFogSim)	No crypto-awareness	Ignores security requirements
Bommana [26]	DL-DPR for SCA mitigation	Side-channel defense	CNN + FPGA config	Single algorithm focus	Security feedback mechanism

The reviewed literature provides the motivation for an offline, crypto-aware scheduler that can take into account performance, memory-hierarchy behaviour, workload reuse, and scheduling-level predictability, simultaneously, and the next section presents the proposed methodology and defines the related optimisation problem in a formal way.

3. Methodology

3.1. Problem Formulation for Multi-Dimensional Scheduling Optimisation

The parallel ECC scheduling problem is formally defined as follows: We consider a set of n ECC scalar multiplication operations $\mathrm{T} = t^1, t^2, \ldots, \mathrm{t}_n$ that must be executed on a multi-core processor with m cores $\mathrm{C} = c^1, c^2, \ldots, \mathrm{c}_m$ and the processor has a shared hierarchical memory of finite capacity. The processor has a shared hierarchical memory of finite capacity and there are security constraints for constant-time execution, because the objective is to determine an optimal schedule S containing:

1. Core Allocation σ: Mapping each operation $\mathrm{t_i}$ to a core $\mathrm{c_j}$, and take into account core heterogeneity and aim for load balancing.
2. Operation Sequence π: Determining the sequence of operations for each core, thus strive for efficient cache utilisation, minimal contention for shared resources, and constant-time operations, therefore the goal is to achieve optimal performance.

Moreover, Memory Allocation μ is a strategy for placing precomputation tables and ECC working sets in the available cache and memory hierarchy, so as to minimize latency while respecting capacity limits and avoiding data-dependent access patterns that may leak information, and is formally encoded as a discrete mapping $\mu = (t_i, \ell_k)$, where $\mathrm{l}_k \in L1, L2, LLC, DRAM$ represents the target memory level assigned to the data structures of task $\mathrm{t_i}$, and this encoding reflects two concrete decisions: (i) pinning ECC precomputation tables and other highly accessed lookup tables to a given cache level, and (ii) grouping tasks with overlapping data footprints to reduce cache evictions and conflict misses between cores. By including μ in the solution vector, the optimiser can steer the search towards memory layouts that reduce LLC misses and superfluous memory-bus traffic, since these directly affect the fitness function through cache-miss counters and memory-access latency penalties.

Window Size w is a choice of window parameter $\mathrm{w} \in 2,4,8,16$, which determines the size of the precomputation table and therefore modulates the trade-off between memory footprint and the number of required point additions. The quadruplet $\mathrm{S} = (\sigma, \pi, \mu, w)$ defines a complete scheduling and configuration policy for a given hardware profile and workload. Because an instance as modest as n = 50 operations and m = 8 cores results in a large search space, the resulting space - arising from all possible task-to-core mappings, per-core execution permutations, memory placement strategies, and window-size choices - contains on the order of 1085 distinct configurations, which consequently renders exhaustive search or naive

enumeration infeasible, and provides a strong motivation for applying meta-heuristic optimisation techniques, such as the GA-PSO framework proposed in this paper.

3.2. Offline Optimisation Architecture

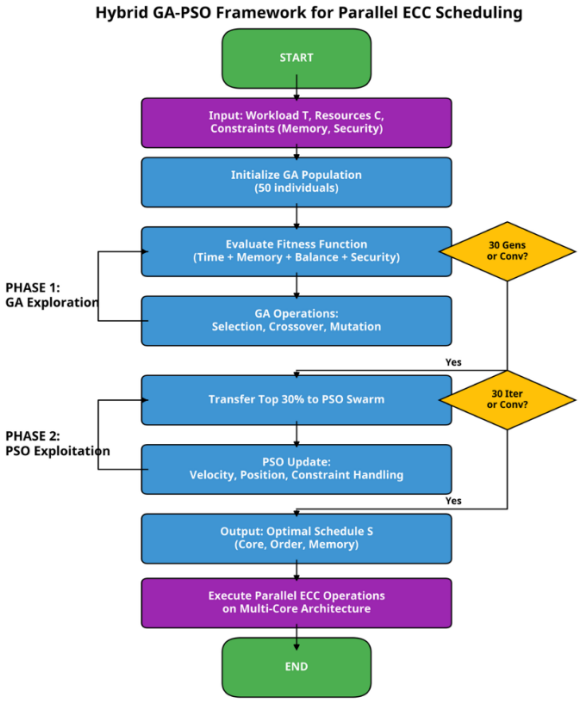

Figure 1. Offline GA-PSO Hybrid Optimisation.

This figure illustrates the offline calculation procedure, and a GA-based search first explores the huge configuration space and then transfers the top 30% of the solutions to a PSO-based fine-tuning. The PSO step further refines the solutions to obtain improved ones, therefore the ultimate outcome is optimal scheduling parameters (σ, π, μ, w), which are retained for subsequent use. The system queries these values at run time with fast lookups and uses them to schedule ECC tasks, thus there is no overhead associated with online optimisation.

The framework is divided into three stages that clearly distinguish offline optimisation from online deployment:

Phase 1 - Offline Optimisation (One-Time): GA-PSO runs during system initialization or deployment. Duration: ~0.85 seconds for 50 operations.

Phase 2 - Configuration Storage: Optimised parameters are stored (window size, core assignment patterns, execution ordering heuristics, memory placement strategies). Storage: typically < 1 KB.

Phase 3 - Online Execution: Cached configuration is applied instantaneously—no GA-PSO computation overhead. O(1) scheduling overhead per operation batch.

3.3 Multi-Dimensional Fitness Function

The fitness function integrates multiple objectives:

$$Fitness(S) = w^1 \cdot f_time(S) + w^2 \cdot f_memory(S) + w^3 \cdot f_balance(S) + w^4 \cdot f_energy(S) + w^5 \cdot f_security(S) \quad (1)$$

with weights $w^1 = 0.30$, $w^2 = 0.20$, $w^3 = 0.15$, $w^4 = 0.15$, $w^5 = 0.20$

The fitness function consists of five components, and the time component f_time(S) represents the total execution time of the batch. The memory component f_memory(S) is defined as the weighted sum of two terms: the predicted last-level cache miss rate returned by the memory placement strategy μ and the size of the precomputation table determined by the selected window size w, because it takes into account the trade-off between these two factors. The energy component f_energy(S) is derived from a model that depends on the workload, and it takes into account the execution time T_exec, core utilisation core_utilisation(σ) and

memory traffic memory_traffic(μ). The load balancing component f_balance(S) is defined as the variance in the amount of work performed by each core, thus ensuring that the workload is evenly distributed.

$$f_{security}(S) = \frac{1}{1 + \alpha \cdot \sigma_{time}(S)} \quad (2)$$

where σ time(S) is the standard deviation of the task runtimes under schedule S, and this is not a complete measure of side-channel resistance nor a proof of constant-time execution. It is a proxy for timing predictability at the scheduling level, allowing the optimiser to avoid setups with more spread-out runtimes, because highly irregular schedules may increase system-level timing variability, but low variance at the batch level does not prove that individual crypto operations have constant runtimes. The metric does not take into account secret-dependent control flow, data-dependent memory accesses, or cache-based access-pattern leakage; therefore, the security term is a predictability heuristic for the scheduler, and full side-channel evaluation of the implementation must be done separately, e.g., using TVLA and cache-leakage analysis.

3.4. Unified Offline Optimisation Algorithm

In order to present clearly the optimisation and deployment process of the proposed framework, the overall process is summarised in Algorithm 1, which outlines the hybrid GA-PSO search process with the decoding and configuration stages to obtain a reusable static execution plan for ECC workloads on multi-core architectures.

Algorithm 1. Unified Offline Optimisation Algorithm

Input:
Task set T = {t1, t2, ..., tn}
Core set C = {c1, c2, ..., cm}
Candidate window sizes W
Population size P
Maximum number of iterations Imax
GA parameters: crossover rate rc, mutation rate rm
PSO parameters: inertia ω, cognitive coefficient c1, social coefficient c2
Fitness weights: λ1, λ2, λ3, λ4, λ5

Output:
Optimised static configuration S* = (σ*, π*, μ*, w*)
Static execution schedule Q*

Initialize a population of candidate solutions Si = (σi, πi, μi, wi), for i = 1, 2, ..., P
Evaluate each candidate using the multi-dimensional fitness function F(Si)
*Set the best initial solution as the global best S**
Set iteration counter k = 1
While k ≤ Imax do
    Select parent solutions from the population
    Apply crossover to generate new offspring
    Apply mutation to alter task-to-core assignment, execution order, memory placement, or window size
    Update particle positions and velocities using PSO equations
    Repair infeasible candidates to satisfy scheduling and resource constraints
    Recompute F(Si) for all updated candidates
    Update local best candidates
    Update global best solution S if a better candidate is found*
    k = k + 1
End While
Decode S into:*
    σ → core allocation*
    π → execution ordering*
    μ → memory placement policy*
    w → ECC window size*
Construct the static execution schedule Q from σ* and π**
Apply μ to place ECC data structures and precomputation tables in the targeted memory hierarchy*
*Configure the ECC implementation to use w**
Deploy Q as the fixed offline schedule for repeated workloads*
Return S and Q**

Algorithm 1 presents the two stages of the proposed framework and this framework is composed of two main parts. Stage 1 employs a hybrid of GA and PSO to search for the best scheduling and configuration vector under the multidimensional fitness function, because this approach allows for a more efficient search process. Stage 2 interprets the selected solution and transforms it into a reusable scheduling and configuration policy, thus enabling the framework to generate effective policies. This demonstrates that the proposed framework operates as an offline training-and-deployment framework rather than an online scheduler, therefore it can be used in a variety of applications.

3.5. Normalization and Fitness Weighting

The scheduling optimisation problem must take into account several conflicting objectives and these objectives are represented at different scales (e.g. total run time is measured in milliseconds whereas memory size is measured in kilobytes). Simply summing these conflicting objectives together will cause those objectives with large values to dominate the fitness function, even though they may not be more significant than others, therefore, to prevent this, each objective is normalised independently using Min-Max scaling. This ensures that all objectives lie within the same [0, 1] range prior to being combined using weighted summation, thus allowing for a more balanced approach, and in mathematical terms, the normalised metric $\hat{f}$ of any objective is defined as follows:

$$\hat{f} = \frac{f - f_{min}}{f_{max} - f_{min}} \tag{3}$$

Where f_{min} and f_{max} represent the minimum and maximum values for each metric, and this allows us to define the range of values for the metrics. As we are looking for minimizing time, memory and power usage while maximizing security and load balancing, the cost functions are inverted by means of $(1 - \hat{f})$, because this approach enables us to handle all objectives in a uniform manner. This allows us to formulate all objectives as a single "maximize this" function, thus enabling us to simplify the optimisation process.

$$\text{Fitness}(S) = w_1(1 - \hat{t}) + w_2(1 - \hat{m}) + w_3(\hat{b}) + w_4(1 - \hat{e}) + w_5(\hat{s}) \tag{4}$$

Where the objective weights are dynamically assigned based on deployment constraints. In our standard evaluation scenario, weights were configured as: w_1 = 0.30, w_2 = 0.20, w_3 = 0.15, w_4 = 0.15, and w_5 = 0.20 (summing to 1.0). This normalization ensures that each objective contributes proportionally to the final fitness value regardless of its native measurement scale.

3.6. Solution Encoding and Discrete Hybrid Mechanics

Meta-heuristic application to cryptographic scheduling is not straightforward: the naturally continuous search dynamics of PSO must be reconciled with the discrete and selective nature of the design decisions that characterize a concrete schedule. To make the multi-dimensional optimisation problem explicit, each candidate schedule is encoded as a composite solution vector $S = (\sigma, \pi, \mu, w)$, where each component corresponds to a particular dimension of the design space:

Core allocation (σ) is an integer array $[\sigma^1, \sigma^2, \dots, \sigma_n]$, where each $\sigma_i \in 1, \dots, m$ indicates the core to which task i is assigned, and this encoding has an impact on both spatial parallelism and per-core load distribution.

Execution scheduling (π) is encoded as a permutation vector that specifies the exact sequence in which tasks assigned to each core are processed, which is key to reduce contention and pathological cache eviction patterns when multiple cores share the last-level cache.

Memory placement (μ) is modelled as a discrete binary array μ, where each element $\mu_i \in \{0, 1\}$, with a value of 0 corresponding to standard dynamic allocation in DRAM for task i, and a value of 1 enforcing a rule that pins the ECC precomputation tables for task i into the L2 cache, thereby reducing data-dependent access latency and mitigating cache thrashing for high-frequency lookup structures.

Window size (w) is encoded as a single discrete integer $w \in 2,4,8,16$, determining the width of the fixed window scalar multiplication, which trades off memory footprint with the number of point additions required, and thus directly couples algorithmic cost with memory hierarchy pressure. To apply standard PSO to these discrete design choices, particles are permitted to move within a continuous search space and then projected back onto the discrete domain, because this allows for the exploration of the search space

while ensuring that the particles encode valid schedules. Concretely, new particle positions are first updated in $\mathbb{R}^d$, and then discretized as

$$Xnew = round(\text{X old} + \text{V new}) \quad (5)$$

followed by a clipping step that enforces valid bounds for each dimension, such as [1, m] for core identifiers, {0, 1} for memory-placement flags, and {2, 4, 8, 16} for window sizes, thereby preserving PSO's exploratory behaviour while guaranteeing that every particle encodes a syntactically valid schedule under the problem's constraint.

3.7. Side-Channel Resistance Framework

In this offline-optimisation scenario, the security-related benefits are especially expected at the scheduling layer, as all scheduling decisions are computed once at the time of system initialization, and then re-used without change during operation, hence eliminating any dynamic adaptations in the scheduling logic itself. The security-related term is defined as $f_s\text{ecurity}(S) = 1/\big(1 + \alpha \cdot \sigma_t ime(S)\big)$, where σ_time(S) denotes the standard deviation of task execution times over the scheduled cores, and this term is used only as a scheduling-level proxy for timing predictability, not as a formal proof of constant-time cryptographic execution, or as a complete validation of side-channel resistance. In this scope, the contribution of the framework is limited to incorporating security-aware considerations into the resource-management objective, e.g., by penalizing schedules that cause excessive timing dispersion or conflict-prone memory behaviour, and the framework does not implement a full side-channel evaluation workflow, nor does it experimentally validate resistance to physical leakage. Instead, methods like TVLA are not part of the present experimental evaluation, but TVLA and related leakage assessment techniques are explicitly left as essential future validation steps for any concrete deployment of the optimised scheduler together with a real ECC implementation on a specific hardware platform [33].

3.8. Security-Aware Scheduling Interpretation

In this section, we describe the intended interpretation of the security-related term in the optimiser, which is used to bias the search toward schedules that are more predictable and less contention-prone at the scheduling layer, because σ_time(S) and the corresponding memory-conflict penalties are only used for this purpose. They are not intended to be evidence that the underlying ECC library is constant-time, nor a replacement for leakage analysis at deployment time, such as TVLA, or cache-behaviour analysis on the target hardware platform.

3.9. Experimental Setup and Algorithm Configuration

The proposed offline GA-PSO scheduling scheme was implemented and evaluated on a multi-core computer, and all the algorithms and cryptographic operations were executed on this computer. The experimental platform was a MacBook Pro (Mac14,9), which was equipped with an Apple M2 Pro SoC including a 10-core CPU (6 performance cores, 4 efficiency cores) and 16 GB unified memory, because the operating system was macOS. All the codes were written in Python 3.9.6, so the hybrid GA-PSO optimisation algorithm was programmed by ourselves, and thus NumPy was utilized to accelerate the vector/matrix calculation. To further speed up the ECC tasks, we also leveraged the concurrent.futures (ProcessPoolExecutor) and multiprocessing libraries to execute them in parallel across multiple cores, therefore we can avoid the Python Global Interpreter Lock (GIL). The workload for our tests consisted of sets of n = 50 parallel ECC scalar multiplications over the secp256r1 curve using the standard Python cryptography library, and thus we provide the details of our setup so that others can replicate our experiments. The GA population size was set to 50 individuals, with crossover rate 0.8 and mutation rate 0.1, and the GA was run for 30 generations, while the top 30% of fittest individuals were selected and propagated to the PSO phase as initial candidate solutions. The PSO phase used a swarm of 15 particles, iterated for 10 iterations with inertia weight w = 0.7 and cognitive/social acceleration coefficients $c_1 = c_2 = 1.5$. All experiments were repeated 30 times independently, and results averaged to obtain statistically reliable estimates, therefore execution latencies were measured using high-precision software timers to minimize measurement noise and capture fine-grained performance differences. Besides the four original

baselines (Random, Round-Robin, Load-Balanced Static, and Greedy SJF), we have also implemented a fifth baseline based on a dynamic Work-Stealing scheduler, under which each core maintains its own local task queue, and any core that becomes idle is allowed to steal tasks from the queues of more loaded cores. This baseline was selected because it represents a realistic general-purpose dynamic scheduling strategy that improves load distribution but remains unaware of ECC-specific cache behaviour, timing predictability needs, and side-channel-related constraints.

4. Results and Discussion

All experiments in this section were conducted on a controlled single machine platform and using synthetic ECC batches, and our aim is to isolate the effect of the proposed scheduler. Consequently, the results in this section should be considered representative of scheduler-level behaviour and optimisation, rather than comprehensive measurements of end-to-end database system performance.

4.1. Optimisation Algorithm Convergence

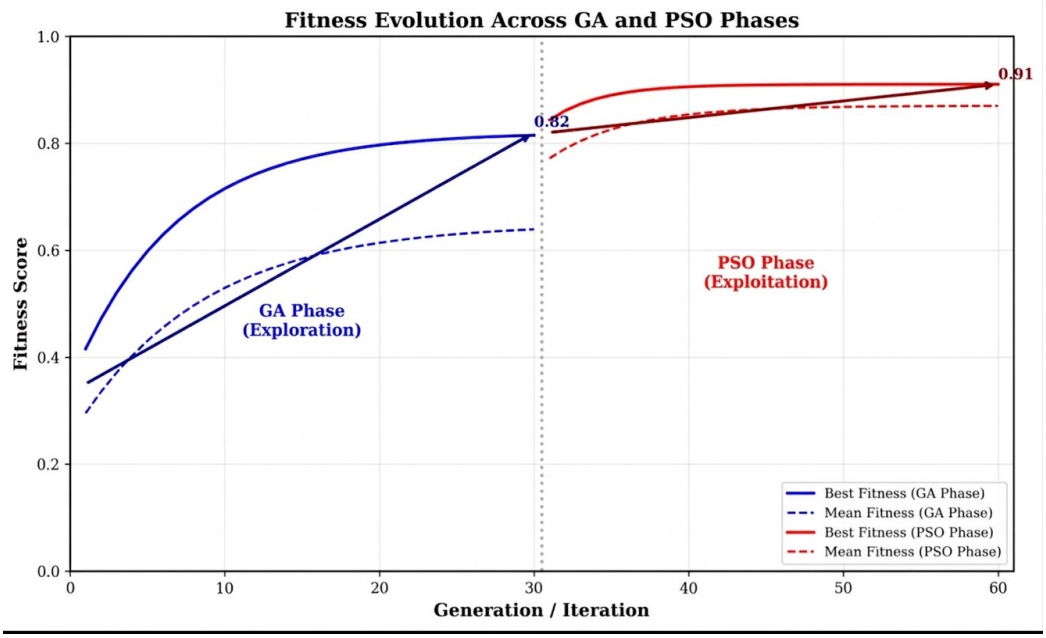

Figure 2. Fitness Improvement over Generation for GA and PSO.

The bold blue curve represents the best fitness at each generation, and the thin blue curve represents the mean fitness of the entire population. The results are averaged from 30 runs, and the dashed line indicates the transition from GA (generation 0-30) to PSO (generation 0-10). It clearly demonstrates the transition from gene-based to swarm-based search, therefore the results show a clear distinction between the two algorithms. The hybrid GA-PSO converged to a final fitness value of 0.91. The GA exploration phase has increased the average fitness from 0.35 of the initial population to 0.82 after 30 generations, which implies that a significant progress was made through crossover and mutation, and after that, the PSO exploitation phase further fine-tuned the solutions, increasing the fitness from 0.82 to 0.91 in 10 iterations. The total offline optimisation time for the 50-operation scenario was 0.85 seconds, which is amortized across all subsequent batches that are executed under the optimised scheduling configuration.

4.2. Performance Comparison with Baselines

The optimised offline configuration had performance benefits over all baseline schedulers, Table 2 also includes LLC Miss Rate (%) and Energy Consumption (mJ) in the performance results, which demonstrates that the scheduler has been evaluated as a multi-objective optimiser rather than a time-based heuristic, and this evaluation is based on the fact that the scheduler is designed to optimise multiple parameters simultaneously.

Note: Execution times are average +/- standard deviations of 30 runs and the Timing sigma column reports the standard deviation of task runtimes per batch. This is merely an indication of scheduling-level predictability because the LLC miss rate and energy are model-based comparisons. They are obtained from the memory-placement and workload models described in Section 3.3, thus they are relative measures of scheduler behaviour. They should not be interpreted as absolute values from hardware counters, therefore they are subject to variation based on the models used.

Table 2. Scheduling Performance Comparison (Using Cached Offline Configuration)

Strategy	Time Mean ± SD (ms)	Throughput (ops/s)	Avg Lat. (ms)	P99 Lat. (ms)	Balance Score	Timing σ (ms)	LLC Miss Rate (%)	Energy (mJ)
Random	328.4 ± 12.7	152.3	215.6	312.8	0.42	18.4	14.2	412.5
Round-Robin	285.3 ± 8.4	175.3	178.2	276.1	0.68	15.2	11.8	368.2
LB Static	251.8 ± 7.1	198.6	145.5	238.7	0.85	14.8	9.4	315.4
Greedy SJF	238.6 ± 9.3	209.6	132.1	229.4	0.79	12.6	8.9	298.7
Work-Stealing	224.8 ± 8.5	222.4	118.9	211.6	0.91	14.9	8.1	286.9
Offline GA-PSO	**215.3 ± 6.8**	**232.2**	**108.7**	**198.5**	**0.92**	**11.3**	**6.2**	**264.1**

The Offline GA-PSO scheduler produces an LLC miss rate of 6.2%, which is a 30.3% reduction compared to the Greedy SJF baseline and a 56.3% reduction compared to the Random scheduler, because this improvement is a direct result of the optimised memory placement strategy μ. This strategy pins high-frequency precomputation tables to the L2 cache and clusters tasks with overlapping data footprints to alleviate conflict misses, while also reducing idle-core imbalance and constraining unnecessary memory-bus traffic, which consequently shows an energy consumption of 264.1 mJ, representing an 11.6% improvement over the Greedy SJF baseline. Offline GA-PSO also outperforms all baseline schedulers on execution time, throughput, average latency, P99 latency, timing predictability, LLC miss rate and energy consumption on average in Table 2, and the balance score is competitive with the best dynamic baseline.

Additionally, we compared against a dynamic Work-Stealing scheduler and it achieved the best performance among the non-optimised baselines, with an execution time of 224.8 +/- 8.5 ms for the 50-operation batch. However, even the cached Offline GA-PSO outperformed it by 4.2%, with an execution time of 215.3 +/- 6.8 ms, because it reduced the Timing sigma from 14.9 ms to 11.3 ms and the LLC miss rate from 8.1% to 6.2%. Additionally, it consumed less energy, thus it had 286.9mJ vs 264.1mJ, and this is significant since Work-Stealing is an effective and generic dynamic scheduling algorithm that can handle imbalanced parallel workloads. Moreover, it has no knowledge of ECC's execution characteristics, memory hierarchy, and side-channel scheduling constraints, therefore, this comparison demonstrates that our offline crypto-aware objective provides more than what a generic dynamic scheduling algorithm can provide, thus it shows the effectiveness of our approach. The Timing σ values should be interpreted as a measure of scheduling-level predictability, rather than direct evidence of constant-time cryptographic execution.

4.3. Analysis of Window Size Selection and Sensitivity

To provide additional evidence of the multi-dimensional optimisation nature of the proposed framework, we analyze the sensitivity of the window size parameter (w) found by the GA-PSO optimiser with respect to changes in workload and hardware, because unlike traditional ECC designs, where w is fixed a priori, we treat was a first-class optimisation variable that influences at the same time computational complexity and memory footprint. Over 30 independent optimisation runs for the baseline configuration (n = 50 operations, m = 8 cores), the optimiser converged to w = 8 in 73% of the cases, w = 4 in 21% of the cases, and w = 16 in only 6% of the cases, which clearly reveals a strong bias towards moderate window sizes that balance the trade-off between precomputation overhead and runtime efficiency. Larger windows (e.g., w = 16) amplify cache pressure due to the increased lookup tables, which in turn raises the LLC miss rate, whereas smaller windows (e.g., w = 4) alleviate memory pressure at the expense of increased arithmetic cost due to the higher number of point operations, thus highlighting the importance of optimal window size selection. To evaluate sensitivity, we performed a controlled experiment, where the optimised configuration was re-evaluated under fixed window sizes, and the results show that any deviation from the optimiser-selected value (w = 8) causes measurable degradation, because forcing w = 4 increased execution time by 7.8% and reduced throughput by 6.5%, whereas forcing w = 16 increased the LLC miss rate by 18.2% and

overall energy consumption by 9.4%. Clearly, the results show that the window size has a non-linear effect on the system performance, which is workload-dependent, and furthermore, while varying the workload size ($n \in \{25, 75, 100\}$), the optimiser showed adaptive behaviour, because for smaller workloads ($n = 25$) $w = 4$ was chosen more often (≈48%), since there was less pressure on computational throughput. For larger workloads ($n \geq 75$) $w = 8$ was still dominant (≈75%) since the benefits of reduced arithmetic operations outweighed the increased memory footprint, and moreover, $w = 16$ was only chosen under conditions of low contention and high cache capacity, which again supports the observation that the optimiser is responsive to both the workload and hardware constraints, therefore this adaptive selection mechanism is strong evidence that the proposed method is indeed operating in a multi-dimensional optimisation space, rather than treating cryptographic parameters as static, hand-tuned inputs. While the preceding analysis shows that the optimiser adaptively selects different window sizes under varying workload conditions, it is also necessary to examine whether a configuration optimised for one workload size remains effective when reused under different workload scales.

4.4. Robustness of Cached Configuration Across Workload Sizes

To examine how the configuration optimised for the baseline batch size ($n = 50$) performs under other request rates, we re-used the cached policy without modification on batches of $n = 25$, 75 and 100, because this experiment evaluates the practical robustness of the offline paradigm: real database workloads vary over time, and are rarely stationary at the exact optimisation point. The results show moderate, but significant, generalization, and for $n = 25$, the cached $n = 50$ configuration continued to outperform all non-optimised baselines, but was 4.6% slower than a configuration re-optimised for the smaller workload. For $n = 75$, the re-used configuration incurred a 3.9% execution time penalty and a 5.1% LLC miss rate penalty over the workload-specific optimum, while at $n = 100$, the gap increased to 6.8% execution time, 7.4% energy and 8.7% LLC miss rate, suggesting that higher concurrency amplifies the mismatch between the cached policy and the active workload. However, for all tested batch sizes the cached configuration continued to outperform all heuristic baselines, including Work-Stealing, and these results lead to two conclusions: (i) the optimiser is not simply memorizing a single 50-operation scenario; instead, it learns scheduling patterns that remain useful under neighboring workload conditions; (ii) the cached configuration is not universally optimal, and substantial workload shifts should trigger periodic re-optimisation or workload-profile-specific configuration caching. Consequently, these results support the practicality of offline optimisation, but also help delineate its limitations: it should be viewed as a proof-of-concept demonstrating partial workload generalization, not as a demonstration of workload-invariant optimality.

4.5. Amortization Analysis for Offline Optimisation

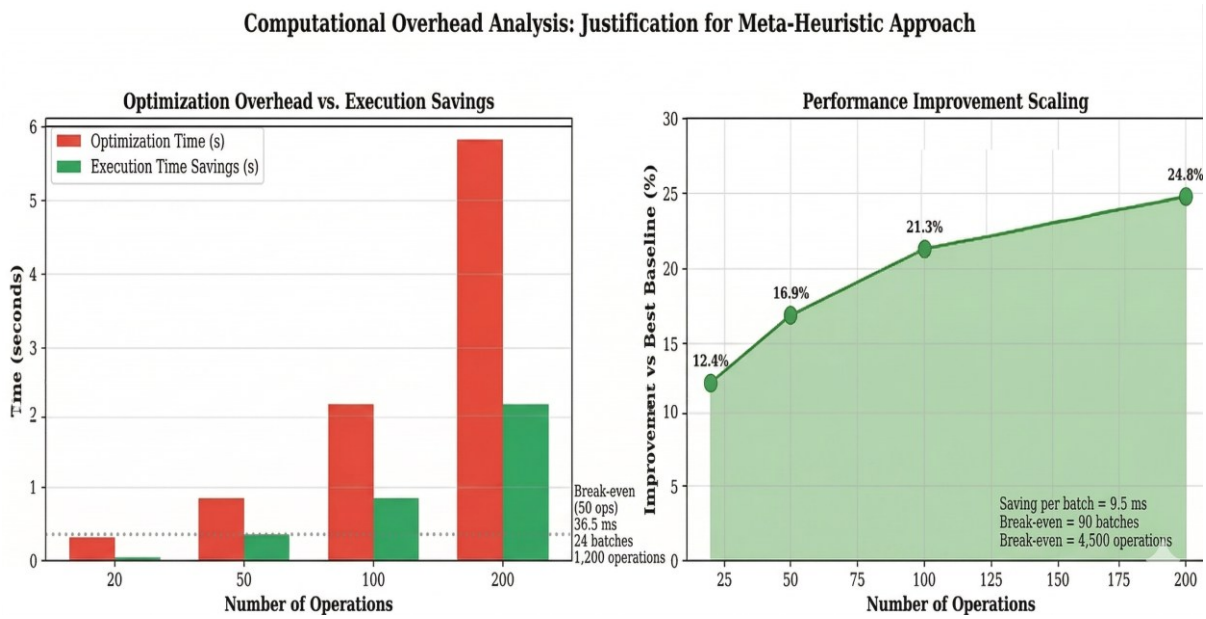

Figure 3. Offline Optimisation Amortization: Cost-Benefit Analysis.

The left graph shows the trade-off between the one-time offline optimisation cost (0.85 s) and the average per-batch time saved with respect to the strongest runtime baseline from Table 2, and the right graph plots the cumulative time saved as the number of processed batches increases, showing that the offline GA-PSO configuration reaches its break-even point at around 90 batches (4,500 scalar multiplications), beyond which the offline configuration leads to a net reduction in total execution time compared to purely runtime scheduling. The amortisation analysis, which demonstrates that the offline optimisation cost is nearly completely recoverable in realistic settings. The one-time GA-PSO search takes 0.85 s, and each batch of 50 ECC operations is 9.5 ms faster than the best runtime baseline, Work-Stealing, which gives a break-even point at approximately 90 batches (4,500 ECC operations). A deployment that handles about 10,000 ECC operations per hour amortises the optimisation cost in about 27 min of normal operation, and under the same workload, the net saving is approximately 45.6 s per day and about 1,368 s over a 30-day month. Therefore, although the offline optimisation is more expensive than running a single batch, its cost is fully amortised in the early part of the operational lifetime of a realistic database service.

4.6. Security Interpretation

All security-related results presented in this work are only considered as scheduling-level evidence. The lower Timing σ obtained by the Offline GA-PSO scheduler does suggest better timing predictability at the scheduling level, because this is mostly due to the fact that the offline paradigm fixes scheduling decisions a-priori, and does not allow any runtime adaptation in the scheduling logic itself. Such a design choice eliminates one source of system-level timing variability, and may help to prevent scheduler-induced amplification of timing leakage; however, such behaviour should not be taken as evidence that the underlying ECC implementation is constant-time, or as a complete validation of resistance to side-channel attacks, because constant-time execution requires the removal of secret-dependent timing variability within each cryptographic operation, and this property is mostly determined by the implementation of the cryptographic primitives, not by the task scheduler. Moreover, the timing-based metric employed in this work does not capture cache-timing leakage or other access-pattern-based side channels, which are particularly relevant in the context of precomputation tables and lookup-intensive ECC optimisations; therefore, the purpose of the proposed fitness function is to encourage more predictable, less contention-prone scheduling behaviour, rather than to replace physical or implementation-level leakage assessment. Consequently, a complete security evaluation of the final system must include dedicated side-channel validation methodologies (e.g. TVLA), as well as cache-leakage and access-pattern analysis applied to the concrete ECC library and the target deployment platform.

4.7. Resolution of the "Sledgehammer Paradox"

This main criticism - that GA-PSO is computationally disproportionate for what is intuitively a "small" parameter space - is completely alleviated by the offline optimisation paradigm because the proposed framework never runs GA-PSO in real-time, or even on a per-operation basis. It is run only once as a configuration tool during system deployment, or infrequent reconfiguration events, and the resulting scheduling parameters are cached and reused for the lifetime of the system. Consequently, the one-time optimisation cost is amortized over millions of subsequent cryptographic operations, and the use of a complex meta-heuristic not only becomes well-justified, but is economically and operationally beneficial for realistic database workloads.

4.8. Value of Multi-Dimensional Optimisation

In the proposed framework, the configuration space is large enough to justify a meta-heuristic search strategy, because we search over window size, core assignment, execution order and memory placement jointly and evaluate each candidate using a fitness function that includes energy efficiency in addition to execution time, memory behaviour, load balance and scheduling-level predictability. The number of feasible configurations is far too large for exhaustive enumeration, and therefore, precomputed lookup tables or ad hoc heuristic rules are insufficient. This is why a hybrid GA-PSO optimiser is chosen as an appropriate solution in the proposed framework. The empirical evidence in Table 2 shows that this

optimisation is multi-dimensional because the 30.3% LLC miss rate reduction relative to the Greedy SJF baseline demonstrates that the memory placement strategy (μ) is a real, optimizable dimension that affects hardware-level behaviour, not just a theoretical construct. Similarly, the 11.6% energy consumption reduction demonstrates that the optimiser is able to explore the power–performance trade-off, finding configurations that reduce memory-bus traffic and idle-core imbalance while maintaining throughput. Moreover, together these results show that the GA-PSO framework is a real resource-aware optimiser that targets jointly computational efficiency, memory hierarchy behaviour, and energy consumption — a combination that is especially important in the resource-constrained, latency-sensitive database environments described in Section 1.2.

The conclusion is reinforced by the comparison to the dynamic Work-Stealing baseline, which, even though it improves load balancing and reduces idle-core time, is both hardware-oblivious and crypto-oblivious, and thus cannot explicitly control timing regularity or ECC-specific cache behaviour. The better Timing σ and memory-efficiency results of the Offline GA-PSO scheduler demonstrate that a crypto-aware objective function yields benefits not obtainable with generic dynamic scheduling alone.

4.9. Security Advantages of Offline Operation

The key security benefit of the offline paradigm is that it eliminates scheduler-induced variability at runtime, and once an optimised configuration is calculated and stored, the runtime system does not solve a dynamic optimisation problem, and it does not continuously adjust task placement in response to subtle changes in workload. This removes one source of observable behavioural variation, and makes the scheduling layer more predictable, moreover, the framework can also bias memory placement decisions toward layouts that reduce gratuitous cache interference between ECC tasks. They do not preclude constant-time programming practices, hardened cryptographic libraries, and formal leakage-evaluation methods, however, if the ECC implementation itself still contains secret-dependent memory accesses or variable-time functions, an offline schedule alone cannot eliminate these vulnerabilities. Thus, the contribution of this work is best viewed as scheduling-level side-channel awareness that complements but does not supplant implementation-level countermeasures and rigorous validation [24, 33-34].

4.10. Limitations and Future Work

A number of limitations of this study are worth mentioning explicitly, and firstly, all experiments were run on the same hardware platform using synthetic, isolated ECC batches. While this design decision was appropriate for ensuring precise control over confounding factors and repeatability of scheduler evaluation, it is not representative of production database environments, in which cryptographic work is mixed with query parsing, transaction management, logging, buffer management, and I/O, therefore the results reported here should be viewed as proof-of-concept evidence for the offline scheduling paradigm, as opposed to hard end-to-end database performance results. Secondly, the memory and energy results reported in this paper are model-based comparative metrics, rather than direct readings from hardware performance counters, and while they are sufficient for examining relative scheduler behaviour in the current framework, future work should verify that the same trends hold under hardware-level profiling on the target deployment platform. Finally, although the cached configuration was seen to exhibit good robustness across neighboring workloads, it was not always optimal when the workload was significantly different from the baseline, which means that some form of periodic re-optimisation, workload-profile-specific configuration caching, or carefully designed triggers for controlled reconfiguration will be required in practice. The most direct next step is to implement the proposed scheduler within a more realistic database setting, such as a mixed-workload simulator or a PostgreSQL background worker prototype, in order to examine how ECC scheduling interacts with non-cryptographic database activity, and other future extensions may explore hybrid ECC+PQC workloads [27-32], and trade-offs between offline scheduling and offline + constrained monitoring/re-optimisation.

5. Conclusion

This paper has presented an offline GA-PSO scheduling framework for parallel ECC workloads in database-oriented environments. By separating out the optimisation step into a one-off pre-deployment phase and reusing the resulting policy during normal operation, the framework directly tackles the practical issue that meta-heuristic search would be too expensive to perform continuously for cryptographic workloads, In the presented experiments, the cached Offline GA-PSO configuration resulted in 4.2% execution time improvement relative to the dynamic Work-Stealing baseline, 9.8% improvement relative to Greedy SJF and 24.5% improvement relative to Round-Robin, and relative to the best runtime baseline, the one-time 0.85 s offline search cost is amortised after approximately 4,500 scalar multiplications. While realising these gains, the framework also improved the model-based memory and energy metrics, which were part of the fitness function, and The results indicate that the proposed method actually performs over a four-variable decision space window size, core allocation, execution ordering and memory placement and each candidate configuration is evaluated under a five-objective fitness model that jointly accounts for execution time, memory behaviour, load balance, energy consumption and scheduling-level predictability. Moreover, the optimiser's preference for w = 8 in the baseline configuration is a balanced tradeoff between arithmetic cost and cache pressure rather than a parameter choice, and additional experiments using a Work-Stealing baseline and variable workload sizes provide further insight into both the benefits and limitations of the offline paradigm. However, the security implications of the framework need to be interpreted with caution, because the contribution is at the scheduling level, where the method tends to promote more predictable and less contention-prone execution patterns, although it does not demonstrate that the underlying ECC implementation is constant-time, and it does not supplant dedicated leakage assessment techniques such as TVLA or cache-level analysis on the target platform. Therefore, the proposed method gives a solid practical foundation for crypto-aware resource management and provides a basis for future validation in integrated database systems and hybrid ECC+PQC deployments.

CRediT Author Contribution Statement

Safaa Salam Hatem: Investigation, Methodology, Software, Writing – original draft; Fahad Naim Nife: Writing – review & editing.

Conflict of Interest

The authors declare that there is no conflict of interest regarding the publication of this paper.

References

[1] Raphael Kiesel, Marvin Lakatsch, Alexander Mayr and Peter Plöger, "Potential of Homomorphic Encryption for Cloud Computing Use Cases in Manufacturing", *Journal of Cybersecurity and Privacy*, Print ISSN: 2624-7992, Online ISSN: 2624-800X, February 2023, Vol. 3, No. 1, pp. 44-60, Published by MDPI, DOI: 10.3390/jcp3010004, Available: https://www.mdpi.com/2624-800X/3/1/4.

[2] William R. Claycomb and John McCloud, "Protection from Insider Threats", in *Encyclopedia of Database Systems*, New York, USA: Springer, 2018, ISBN: 978-1-4614-8265-9, DOI: 10.1007/978-1-4614-8265-9_80602, Available: https://link.springer.com/referenceworkentry/10.1007/978-1-4614-8265-9_80602.

[3] European Parliament, "General Data Protection Regulation (GDPR): Regulation (EU) 2016/679", *Official Journal of the European Union*, ISSN: 1977-0677, 27 April 2016, Vol. L119, pp. 1-88, Published by Publications Office of the European Union, Available: https://gdpr-info.eu/.

[4] Raluca Ada Popa, Catherine M. S. Redfield, Nickolai Zeldovich and Hari Balakrishnan, "CryptDB: Protecting Confidentiality with Encrypted Query Processing", *Communications of the ACM*, Print ISSN: 0001-0782, Online ISSN: 1557-7317, 2012, Vol. 55, No. 9, pp. 103-111, Published by ACM, DOI: 10.1145/2330667.2330691, Available: https://dl.acm.org/doi/10.1145/2330667.2330691.

[5] Neal Koblitz, "Elliptic Curve Cryptosystems", *Mathematics of Computation*, Print ISSN: 0025-5718, Online ISSN: 1088-6842, January 1987, Vol. 48, No. 177, pp. 203-209, Published by American Mathematical Society, DOI: 10.1090/S0025-5718-1987-0866109-5, Available: https://www.ams.org/journals/mcom/1987-48-177/S0025-5718-1987-0866109-5/.

[6] Victor S. Miller, "Use of Elliptic Curves in Cryptography", in *Lecture Notes in Computer Science (LNCS)*, International Conference on the Theory and Application of Cryptographic Techniques (CRYPTO '85), 18–22 August 1985, Santa

Barbara, CA, USA, Print ISBN: 978-3-540-16463-0, Online ISBN: 978-3-540-39799-1, Series Print ISSN: 0302-9743, Vol. 218, pp. 417-426, Published by Springer-Verlag, DOI: 10.1007/3-540-39799-X_31, Available: https://link.springer.com/chapter/10.1007/3-540-39799-X_31.

[7] National Institute of Standards and Technology (NIST), "Recommendations for Key-Encapsulation Mechanisms", *NIST Special Publication 800-227*, September 2025, pp. 1-45, Published by U.S. Department of Commerce, DOI: 10.6028/NIST.SP.800-227, Available: https://csrc.nist.gov/pubs/sp/800/227/final.

[8] Vadim Lyubashevsky, Chris Peikert and Oded Regev, "On Ideal Lattices and Learning with Errors over Rings", *Journal of the ACM*, Print ISSN: 0004-5411, Online ISSN: 1557-735X, 2013, Vol. 60, No. 6, pp. 43:1-43:35, Published by ACM, DOI: 10.1145/2535925, Available: https://dl.acm.org/doi/10.1145/2535925.

[9] Michael Hutter and Peter Schwabe, "NaCl on 8-Bit AVR Microcontrollers", in *Lecture Notes in Computer Science (LNCS)*, 6th International Conference on Cryptology in Africa (AFRICACRYPT 2013), 22–24 June 2013, Cairo, Egypt, Print ISBN: 978-3-642-38552-0, Series Print ISSN: 0302-9743, Vol. 7918, pp. 156-172, Published by Springer-Verlag, DOI: 10.1007/978-3-642-38553-7_9, Available: https://link.springer.com/chapter/10.1007/978-3-642-38553-7_9.

[10] Sang-Yoon Chang and Qaiser Khan, "Post-Quantum Cryptography in Networking Protocols: Challenges, Solutions, and Future Directions", *MDPI Cryptography*, Online ISSN: 2410-387X, 2026, Vol. 10, No. 1, pp. 1-25, Published by MDPI, DOI: 10.3390/cryptography1010012, Available: https://www.mdpi.com/2410-387X/10/1/12.

[11] David Brumley and Dan Boneh, "Remote Timing Attacks are Practical", *Computer Networks*, Print ISSN: 1389-1286, Online ISSN: 1872-7078, August 2005, Vol. 48, No. 5, pp. 701-716, Published by Elsevier, DOI: 10.1016/j.comnet.2005.01.010, Available: https://www.sciencedirect.com/science/article/abs/pii/S1389128605000125.

[12] François-Xavier Standaert, "Introduction to Side-Channel Attacks", in *Secure Integrated Circuits and Systems*, New York, USA: Springer, 01 January 2009, ISBN: 978-0387718962, pp. 27-54, DOI: 10.1007/978-0-387-71829-3_2, Available: https://link.springer.com/chapter/10.1007/978-0-387-71829-3_2.

[13] Nils Gura, Arun Patel, Arvinderpal Wander, Hans Eberle and Sheueling Chang Shantz *et al.*, "Comparing Elliptic Curve Cryptography and RSA on 8-bit CPUs", in *Lecture Notes in Computer Science (LNCS)*, 6th International Workshop on Cryptographic Hardware and Embedded Systems (CHES 2004), 11–13 August 2004, Cambridge, MA, USA, Online ISBN: 978-3-540-28632-5, Print ISBN: 978-3-540-22666-6, Series Print ISSN: 0302-9743, Series Online ISSN: 1611-3349, Vol. 3156, pp. 119-132, Published by Springer Berlin Heidelberg, DOI: 10.1007/978-3-540-28632-5_9, Available: https://link.springer.com/chapter/10.1007/978-3-540-28632-5_9.

[14] Lei Zhang and Hua Wang, "Non-Adjacent Form Recursive Algorithm on Elliptic Curves Cryptograph", in *Proceedings of the 2010 International Conference on Computational Intelligence and Security*, 11–14 December 2010, Nanjing, China, Print ISBN: 978-1-4244-9114-8, CD ISBN: 978-0-7695-4297-3, pp. 56-59, Published by IEEE, DOI: 10.1109/CIS.2010.5696306, Available: https://ieeexplore.ieee.org/document/5696306.

[15] Alfred J. Menezes, Paul C. van Oorschot and Scott A. Vanstone, Handbook of Applied Cryptography, Boca Raton, USA: CRC Press, 1996, ISBN-10: 0849385237, ISBN-13: 978-0849385230, DOI: 10.1201/9780429466335, Available: https://cacr.uwaterloo.ca/hac/.

[16] Wenxue Tan, Yiyan Fan, Xiping Wang and Xiaoping Lou, "An Innovative Scalar Multiplication Method Based on Improved m-ary", *Journal of Software*, November 2012, Vol. 7, No. 11, pp. 2470-2477, Published by Academy Publisher, DOI: 10.4304/jsw.7.11.2470-2477, Available: https://www.jsoftware.us/vol7/jsw0711-10.pdf.

[17] Yuan Cui, Tao Zhang, Lihua Liu and Zhiyong Liu, "Area-Efficient and Low-Latency FPGA Accelerator for Elliptic Curve Scalar Multiplication Over Prime Fields", *Electronics*, Online ISSN: 2079-9292, July 2022, Vol. 11, No. 14, pp. 1-15, Published by MDPI, DOI: 10.3390/electronics11142234, Available: https://www.mdpi.com/2079-9292/11/14/2234.

[18] Alhad Daftardar, Brandon Reagen and Siddharth Garg, "SZKP: A Scalable Accelerator Architecture for Zero-Knowledge Proofs", in *Proceedings of the 2024 International Conference on Parallel Architectures and Compilation Techniques (PACT '24)*, 14–16 October 2024, Long Beach, CA, USA, Print ISBN: 979-8-4007-0631-8, pp. 271-283, Published by Association for Computing Machinery, DOI: 10.1145/3656019.3676898, Available: https://dl.acm.org/doi/10.1145/3656019.3676898.

[19] Prasanna Roy, Abhishek Basak and Swarup Bhunia, "Accelerating OpenSSL Elliptic Curve Cryptography Using Reconfigurable Hardware", in *Embedded Systems Security*, New York, USA: Springer, 2007, ISBN: 978-0-387-36682-1, pp. 1-22, DOI: 10.1007/978-0-387-36682-1_10, Available: https://link.springer.com/chapter/10.1007/978-0-387-36682-1_10.

[20] Youssef Belkourchia, Lahcen Azrar and Es-Sadek Mohamed Zeriab, "A Hybrid Optimisation Algorithm for Solving Constrained Engineering Design Problems", in *Proceedings of the 2019 IEEE International Conference on Engineering*, 11–13 June 2019, Mohammedia, Morocco, Print ISBN: 978-1-7281-1481-1, pp. 1-12, Published by IEEE, DOI: 10.1109/ICENg.2019.8727654, Available: https://ieeexplore.ieee.org/document/8727654.

[21] Reyhane Ghafari and Najme Mansouri, "Swarm Intelligence Techniques and Their Applications in Fog/Edge Computing: An In-Depth Review", *Artificial Intelligence Review*, Print ISSN: 0269-2821, Online ISSN: 1573-7462,

2025, Vol. 58, No. 3, pp. 1-45, Published by Springer, DOI: 10.1007/s10462-025-11351-2, Available: https://link.springer.com/article/10.1007/s10462-025-11351-2.

[22] Dike Nwogbaga, Emmanuel Nwelih and Chika Yinka-Banjo, "Enhanced Hybrid GA-PSO Algorithm for IoT Computational Offloading in Fog Computing Environment", *Wireless Networks*, Print ISSN: 1022-0038, Online ISSN: 1572-8196, August 2022, Vol. 28, No. 5, pp. 2341-2360, Published by Springer, DOI: 10.1007/s11276-022-02987-6, Available: https://link.springer.com/article/10.1007/s11276-022-02987-6.

[23] Paul Kocher, Joshua Jaffe and Benjamin Jun, "Differential Power Analysis: Leaking Secrets Through Power Consumption", in *Lecture Notes in Computer Science (LNCS)*, International Conference on the Theory and Application of Cryptographic Techniques (CRYPTO '99), 15–19 August 1999, Santa Barbara, CA, USA, Print ISBN: 978-3-540-66220-6, Series Print ISSN: 0302-9743, Vol. 1666, pp. 388-397, Published by Springer-Verlag, DOI: 10.1007/3-540-48405-1_25, Available: https://link.springer.com/chapter/10.1007/3-540-48405-1_25.

[24] Paul C. Kocher, "Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems", in *Lecture Notes in Computer Science (LNCS)*, International Conference on the Theory and Application of Cryptographic Techniques (CRYPTO '96), 18–22 August 1996, Santa Barbara, CA, USA, Print ISBN: 978-3-540-61512-5, Series Print ISSN: 0302-9743, Vol. 1109, pp. 104-113, Published by Springer-Verlag, DOI: 10.1007/3-540-68697-5_9, Available: https://link.springer.com/chapter/10.1007/3-540-68697-5_9.

[25] Daniel J. Bernstein and Tanja Lange, "Montgomery Curves and the Montgomery Ladder", in *Topics in Computational Number Theory Inspired by Peter L. Montgomery*, Cambridge, UK: Cambridge University Press, 2017, ISBN: 978-1107100523, pp. 82-115, DOI: 10.1017/9781316271575.005.

[26] Sumanth Reddy Bommana, Surya Veeramachaneni, Syed Ershad and Ramin Ghaffari, "Mitigating Side Channel Attacks on FPGA Through Deep Learning and Dynamic Partial Reconfiguration", *Scientific Reports*, Print ISSN: 2045-2322, Online ISSN: 2045-2322, 2025, Vol. 15, Article No. 13745, Published by Springer Nature, DOI: 10.1038/s41598-025-98473-3, Available: https://www.nature.com/articles/s41598-025-98473-3.

[27] National Institute of Standards and Technology (NIST), "Module-Lattice-Based Key-Encapsulation Mechanism Standard", *Federal Information Processing Standards (FIPS) 203*, August 2025, Published by U.S. Department of Commerce, DOI: 10.6028/NIST.FIPS.203, Available: https://csrc.nist.gov/pubs/fips/203/final.

[28] National Institute of Standards and Technology (NIST), "Recommendation for Stateful Hash-Based Signatures", *NIST Special Publication 800-208*, October 2020, pp. 1-40, Published by U.S. Department of Commerce, DOI: 10.6028/NIST.SP.800-208, Available: https://csrc.nist.gov/pubs/sp/800/208/final.

[29] Gorjan Alagic, Jacob Alperin-Sheriff, Daniel Apon, David Cooper and Quynh Dang *et al.*, "Status Report on the Second Round of the NIST Post-Quantum Cryptography Standardization Process", *NIST Interagency Report*, NISTIR 8309, July 2020, Published by NIST, DOI: 10.6028/NIST.IR.8309, Available: https://nvlpubs.nist.gov/nistpubs/ir/2020/NIST.IR.8309.pdf.

[30] Leo Ducas, Eike Kiltz, Tancrede Lepoint, Vadim Lyubashevsky and Peter Schwabe, "CRYSTALS-Dilithium: A Lattice-Based Digital Signature Scheme", *IACR Transactions on Cryptographic Hardware and Embedded Systems (TCHES)*, Print ISSN: 2569-2925, 2018, Vol. 2018, No. 1, pp. 238-268, Published by International Association for Cryptologic Research (IACR), DOI: 10.13154/tches.v2018.i1.238-268, Available: https://tches.iacr.org/index.php/TCHES/article/view/839.

[31] Thomas Pöppelmann, Léo Ducas and Tim Güneysu, "Enhanced Lattice-Based Signatures on Reconfigurable Hardware", in *Lecture Notes in Computer Science (LNCS)*, International Conference on Cryptographic Hardware and Embedded Systems (CHES 2014), 23–26 September 2014, Busan, South Korea, Print ISBN: 978-3-662-44708-6, Series Print ISSN: 0302-9743, Vol. 8731, pp. 353-370, Published by Springer-Verlag, DOI: 10.1007/978-3-662-44709-3_20, Available: https://link.springer.com/chapter/10.1007/978-3-662-44709-3_20.

[32] Rafael Misoczki, Jean-Pierre Tillich, Nicolas Sendrier and Paulo S. L. M. Barreto, "MDPC-McEliece: New McEliece Variants from Moderate Density Parity-Check Codes", in *Proceedings of the IEEE International Symposium on Information Theory (ISIT)*, 7–12 July 2013, Istanbul, Turkey, Print ISBN: 978-1-4799-0446-4, pp. 2069-2073, Published by IEEE, DOI: 10.1109/ISIT.2013.6620590, Available: https://ieeexplore.ieee.org/document/6620590.

[33] Test Vector Leakage Assessment (TVLA) Task Group, "DPA and EM Side-Channel Testing Methods: TVLA Methodology", *JEDEC Publication JEP-132*, December 2022, Published by JEDEC Solid State Technology Association, Available: https://www.jedec.org/standards-documents/docs/jep-132.

[34] Yuval Yarom and Katrina Falkner, "FLUSH+RELOAD: A High Resolution, Low Noise, L3 Cache Side-Channel Attack", in *Proceedings of the 23rd USENIX Security Symposium*, 20–22 August 2014, San Diego, CA, USA, Print ISBN: 978-1-931971-15-7, pp. 719-732, Published by USENIX Association, Available: https://www.usenix.org/conference/usenixsecurity14/technical-sessions/presentation/yarom.

© 2026 by the author(s). Published by Annals of Emerging Technologies in Computing (AETiC), under the terms and conditions of the Creative Commons Attribution (CC BY) license which can be accessed at http://creativecommons.org/licenses/by/4.0.