Research Article

Lorenzo Gasparini, Elia Onofri*, Martina Palmucci and Marco Pedicini

Cross-primitive comparison in CP-ABE

Bilinear pairing versus lattices

https://doi.org/10.1515/jmc-2025-0051
Received November 17, 2025; accepted December 17, 2025; published online March 16, 2026

Abstract: Ciphertext-Policy Attribute-Based Encryption (CP-ABE) is a cornerstone technique for fine-grained access control in encrypted data sharing, with pairing-based schemes having established the *de facto* standard. However, their vulnerability to quantum attacks generated growing interest in alternatives rooted in quantum-secure primitives, with Learning With Errors (LWE) emerging as a prominent candidate. In this work, we present the first direct performance benchmark of lattice-versus pairings-based CP-ABE implementations. We compare two mature, widely adopted implementations – PALISADE-ABE's LWE-based Zhang-Zhang scheme and OpenABE's pairing-based Waters scheme – selected for their comparable development environments and assumptions. Our results highlight the practical trade-offs – both costs and benefits – of each paradigm, quantifying where lattice-based designs offer advantages and where they incur extra overhead. This comparison addresses a critical gap in the literature and provides direction for the design of future post-quantum secure systems.

Keywords: CP-ABE; Attribute-Based Encryption; benchmarking; OpenABE; Palisade

MSC 2020: 68P25; 94A60; 68Q25; 94A62

1 Introduction

In recent years, Ciphertext-Policy Attribute-Based Encryption (CP-ABE) has attracted growing attention for its suitability in enforcing fine-grained access control over encrypted data, particularly in cloud-based environments. Since the seminal work by Bethencourt, Sahai, and Waters [1], most schemes proposed in the literature have relied on pairing-based cryptography, which has become the standard foundation for CP-ABE. This is reflected in the breadth of available implementations and benchmarking studies, such as the comprehensive survey in [2].

By contrast, alternative approaches – including those based on elliptic curves [3] and lattice-based cryptography [4] – remain comparatively underexplored. Pairing-based schemes depend on hardness assumptions such as the Decisional Diffie–Hellman (DDH) problem [5, 6], which is known to be vulnerable to quantum algorithms [7], therefore, it is particularly worth finding new schemes.

***Corresponding author: Elia Onofri**, Computer, Electrical and Mathematical Sciences and Engineering (CEMSE), King's Abdullah University of Science and Technology (KAUST), Thuwal, Saudi Arabia; and Istituto per le Applicazioni del Calcolo (IAC), National Research Council of Italy (CNR), Naples, Italy, E-mail: elia.onofri@kaust.edu.sa. https://orcid.org/0000-0001-8391-2563
Lorenzo Gasparini and Marco Pedicini, Department of Mathematics and Physics, Roma Tre University, Rome, Italy.
https://orcid.org/0000-0002-9016-074X (M. Pedicini)
Martina Palmucci, NTT DATA Italia S.p.a., Rome, Italy

Table 1: Overview of CP-ABE schemes implemented in major open-source libraries.

Algorithm	Primitive	Library (language)						Ref.
		OpenABE (C++) [11]	Charm (Python) [13]	Rabe (Rust) [14]	CiFer (C) [15]	GoFe (Go) [16]	PALISADE (C++) [9, 10]	
WATERS	Pairing	■	■					[12]
JYJGXD	Pairing		■					[17]
RW	Pairing		■					[18]
YAHK	Pairing		■					[19]
CGW	Pairing		■					[20]
FAME	Pairing		■		■	■		[21]
AR	Pairing		■					[22]
TBPRE	Pairing		■					[23]
YLLC	Pairing		■					[24]
AW	Pairing		■	■				[25]
BSW	Pairing		■	■				[1]
BDABE	Pairing			■				[26]
ZZ	Lattices						■	[4]

The prospect of practical quantum computers calls for a broader investigation of cryptographic primitives that can offer post-quantum security guarantees. Among these, lattice-based constructions are considered especially promising due to their strong theoretical foundations and versatility [8].

Despite the urgency of transitioning to quantum-resistant solutions, the literature currently lacks direct and systematic performance comparisons between pairing-based and lattice-based CP-ABE implementations. This paper contributes to filling this gap by presenting a benchmarking study that evaluates both approaches under realistic conditions, with attention to efficiency, scalability, and implementation maturity.

Our analysis focuses on well-established open-source libraries that are actively developed and validated by the cryptographic community (see Table 1). For what concerns the lattice-based setting, we consider PALISADE-ABE [9], a C++ library derived from the PALISADE project [10], which was integrated into the OpenFHE framework. PALISADE-ABE includes an implementation of the 2011 CP-ABE scheme proposed by Zhang and Zhang [4]. To ensure a consistent basis for comparison in terms of implementation language, we selected OpenABE [11] as the representative pairing-based library, being one of the most established and feature-rich frameworks for attribute-based encryption, supporting a variety of schemes including the construction from [12] by Waters.

1.1 Related work

Originally introduced by Sahai and Waters in [27] as a fuzzy generalisation of the identity-based encryption (IBE, see [28]), Attribute-Based Encryption (ABE) is an asymmetric “one-to-many” encryption paradigm, offering fine-grained access control over encrypted data through the paired use of attributes and access policies. In other words, ABE is a class of Public Key Encryption (PKE) schemes where decryption of a ciphertext is allowed only when the set of available attributes is compliant with the chosen policy. Due to this feature, ABE has gained considerable interest in the recent literature [29–31], particularly in cloud-based scenarios where a single encryption suffices for any large set of recipients.

Due to the dual role of attributes and policies, two main variants of ABE have emerged: Key-Policy ABE (KP-ABE), where policies are bound to secret keys, and CP-ABE, where they are instead associated with ciphertexts. While KP-ABE was first formalised in 2006 by Goyal et al. [32], CP-ABE model – introduced in 2007 by Bethencourt et al. [1] – has become more widely adopted due to its greater flexibility in expressing access policies.

Over the years, bilinear pairings have become the dominant cryptographic primitive for constructing ABE systems, primarily due to their expressive capabilities and manageable efficiency. Noteworthy milestones include the fully secure functional encryption scheme by Sahai et al. [33], the efficient and expressive CP-ABE

system by Waters [12], and the dual-system encryption framework for prime-order groups by Attrapadung [34], grounded in the formalism introduced in [35]. For a comprehensive overview of the evolution of pairing-based ABE, we refer the reader to the recent survey by Venema et al. [36].

Benchmarking efforts have thus far mostly concentrated on pairing-based ABE schemes. For instance, De la Piedra et al. [37] proposed "ABE Squared", a benchmarking framework for evaluating the efficiency of ABE implementations, while Mosteiro-Sanchez et al. [38] provided a comparative study of existing ABE libraries from a developer-oriented perspective. However, these analyses focus exclusively on pairing-based constructions, overlooking schemes based on alternative cryptographic assumptions.

A similar imbalance is observed in the landscape of open-source CP-ABE implementations, summarised in Table 1. Libraries such as Charm [13], Rabe [14], and OpenABE [11] support a wide array of pairing-based constructions. By contrast, to the best of our knowledge, PALISADE-ABE [9, 10] stands as the only publicly available and maintained library implementing lattice-based CP-ABE schemes.

In particular, the growing menace posed by the advent of quantum computing [39] further accentuates this disparity, casting shadows on primitives like pairing due to their intrinsic weakness to post-quantum attacks, like Shor's algorithm [7]. In contrast, lattice-based cryptography is regarded as a strong candidate for post-quantum security [40], with hardness assumptions based on problems such as Short Independent Vector Problem (SIVP), approximate Shortest Vector Problem (gapSVP), and Learning With Errors (LWE) [41].

Despite this premise, only a limited number of lattice-based CP-ABE schemes can be found in the literature: one of the earliest being the scheme by Zhang and Zhang [4], based on LWE, later extended to support threshold policies and multi-valued attributes in [42]. In 2017, a variant leveraging Ring-LWE was proposed by Chen et al. [43], while Cianfriglia et al. [44] presented a server-mediated extension in 2024.

A significant leap in expressiveness was achieved by Datta et al. [45], who constructed the first CP-ABE scheme over LWE supporting access policies in NC^1 circuits (later expanded in [46, 47]). Also, schemes based on refined variants of LWE have recently gained traction; as an example, Wee [48] introduced a construction for unbounded-size circuits using tensor and evasive-LWE assumptions, and Waters et al. [49] proposed a multi-authority ABE for subset policies using a concretely instantiated evasive-LWE-based random oracle. More recently, Hsieh et al. introduced in [50] a general framework for CP-ABE via evasive inner-product functional encryption, and Wee [51] proposed an almost optimal circuit-supporting construction under succinct LWE.

Despite these theoretical advances, implementations of modern lattice-based ABE schemes remain scarce. This is due in part to the greater complexity in realising such schemes, as well as the difficulty of identifying suitable parameters that balance security and efficiency. For instance, the scheme proposed in [42] lacks a practical implementation due to its high computational requirements in practice.

1.2 Paper contribution and organisation

This work presents a comparative study of CP-ABE schemes grounded on bilinear pairings and lattices. While both primitives are established in the literature and supported by mature open-source libraries, our primary objective is to investigate the broader implications of the underlying primitives – particularly in light of the growing relevance of post-quantum cryptography.

To this end, we consider two representative implementations: the pairing-based scheme by Waters [12], as available in the OpenABE library [11], and the lattice-based scheme by Zhang and Zhang [4], implemented in the PALISADE-ABE library [9, 10]. These libraries have been selected based on their active development status, community validation, and completeness in supporting real-world attribute-based encryption workflows.

The analysis is structured as a benchmark campaign covering the main cryptographic operations involved in CP-ABE – namely, setup, key generation, encryption, and decryption – under a consistent set of parameters and policies. This enables a systematic comparison of computational performance and practical deployability across the two settings.

By offering a reproducible evaluation framework and reporting empirical results, this study aims to provide insight into the trade-offs associated with adopting quantum-resistant primitives in the context of fine-grained access control. To the best of our knowledge, this constitutes the first benchmark contrasting pairing-based and

lattice-based approaches to CP-ABE, thus contributing to the broader understanding of their respective potential in future cryptographic systems.

The remainder of the paper is organised as follows. Section 2 introduces the needed background on CP-ABE. Section 3 describes the methodology adopted for our benchmark campaign; this includes the design of test cases (Section 3.1), the chosen security levels (Section 3.2), attribute and policy configurations (Section 3.3), and the performance metrics used to assess the schemes (Section 3.4). Section 4 presents the results obtained for the four fundamental CP-ABE operations: setup (Section 4.1), key generation (Section 4.2), encryption (Section 4.3), and decryption (Section 4.4); each operation is analysed comparatively in terms of efficiency and scalability. Section 5 concludes the paper by summarising the contribution and discussing future works. Finally, the interested reader can find an introductory overview of the bilinear pairings setting and the lattice-based environment in Appendices A and B, respectively: a short description of the two algorithms used in the benchmark is contextually provided.

2 The CP-ABE framework

In what follows, we provide a more detailed description of ABE, mainly focusing on the Ciphertext-Policy variant.

A CP-ABE system involves three main entities: (i) a Central Authority (CA), (ii) one or more data owners, and (iii) the users. In short (see also Figure 1), the CA is responsible for initialising the system and generating a pair of master keys: a public key *mpk* used for encryption and a master secret key *msk* used to issue personalised user keys. Upon successful enrolment, users receive a private key *sk* associated with a specific set of attributes S. Data owners encrypt plaintext messages M under an access policy W, using the public key *mpk*. Notably, encryption can be performed without interacting with the CA and does not require the data owner to possess a secret key, analogous to traditional public-key encryption schemes. Finally, a user is able to decrypt a ciphertext C only if the attribute set S embedded in their key satisfies the access policy W, denoted as $S \vdash W$. Otherwise, the decryption fails, and the user is unable to recover the original message.

Formally, a CP-ABE scheme is defined by four core algorithms:

Setup(λ, U) $\rightarrow$ (msk, mpk). Executed by the CA, this algorithm takes as input the security parameter λ and a predefined universe of attributes U. It outputs the master secret key *msk* and the public key *mpk*. The former is retained by the CA for issuing user keys; the latter is distributed to enable encryption.

KeyGeneration(msk, S) $\rightarrow$ sk. Also run by the CA, this algorithm generates a secret key *sk* for a user characterised by the attribute set $S \subseteq U$. Implicitly, this represents the point of entry of novel users in the system.

Encryption(mpk, W, M) $\rightarrow$ C. This algorithm allows a data owner to encrypt a message M under an access structure (policy) W, using the public key *mpk*. Since no secret credentials are required for encryption, as in regular PKE, this operation is non-interactive and does not necessitate enrollment.

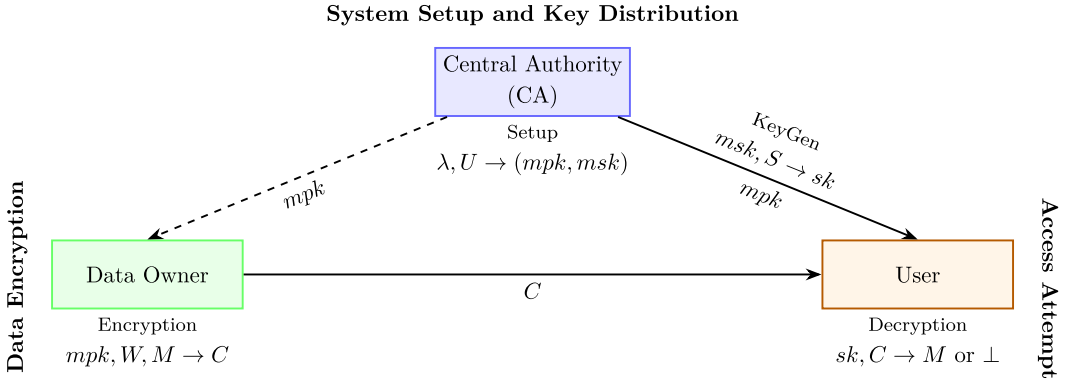

Figure 1: High-level structure of a CP-ABE system. The CA setups the system and issues keys, data owners encrypt with public parameters, and users decrypt based on attribute-policy satisfaction.

Decryption(sk, C) $\rightarrow M \cup \perp$. Executed by a user, this algorithm attempts to decrypt a ciphertext C using the secret key sk. Decryption succeeds and returns M if the attribute set S encoded in sk satisfies the access policy W (*i.e.*, $S \vdash W$). Otherwise, the output is $\perp$, typically corresponding to an unintelligible message.

Numerous extensions of the classical CP-ABE model have been proposed in the literature to enhance its expressiveness and applicability. These include mechanisms for key escrow and recovery [52], key revocation [44], and multi-authority settings [53].

For the purposes of this work, it is particularly relevant to highlight the distinction between the "small universe" and the enhanced "large universe" constructions. In the classical small-universe schemes, the CA defines the finite set of admissible attributes U at setup, which constrains future key and policy generation. Any change in U requires reinitialisation of the system. In contrast, large-universe schemes allow attributes to be drawn from an unbounded domain at any time, without the need to redefine the system. The pairing-based scheme by Waters [12], which we consider in our evaluation, naturally supports the large-universe model.

3 Methods

In this section, we detail parameters and configurations adopted in our benchmarking setup, providing the rationale behind each design decision. Our goal is to ensure a fair and informative comparison by aligning the benchmark conditions as closely as possible, while accounting for the intrinsic differences between the pairing-based and lattice-based cryptographic settings.

To this end, we select two open-source libraries: PALISADE-ABE [9] and OpenABE [11]. Both are implemented in native C++, simplifying direct performance comparisons. OpenABE is built atop the RELIC [54] cryptographic toolkit, which provides support for elliptic curve and pairing-based operations. By contrast, PALISADE-ABE leverages the PALISADE-core library (now maintained within OpenFHE), which is designed for efficient lattice-based computations, including fully homomorphic encryption.

The choice of PALISADE-ABE is essentially dictated by its status as the only publicly available library offering a lattice-based implementation of CP-ABE. OpenABE, on the other hand, is chosen due to its programming language compatibility and its reputation for being one of the most efficient and flexible implementations of pairing-based ABE, as supported by prior evaluations [38].

Design differences. Several fundamental distinctions arise between the two schemes (see also Table 2):

- **Attribute universe.** OpenABE supports a large universe construction, allowing attributes to be defined dynamically. In contrast, PALISADE adopts a small universe model with attributes fixed at setup. As a result, OpenABE benefits from greater flexibility and more compact public parameters.
- **Attribute semantics.** PALISADE enables attributes to be explicitly positive or negative, enforcing semantic constraints such as the impossibility of possessing both a and $\neg a$. OpenABE models attributes as multi-valued key-value pairs, which subsume PALISADE's binary logic while allowing for more expressive encoding.
- **Access policy expressiveness.** OpenABE implements access policies as general Boolean formulas supporting conjunctions, disjunctions, and threshold gates. PALISADE supports only conjunctions over signed attributes. While this limits expressiveness, it enables PALISADE to more rigorously distinguish between

Table 2: Qualitative comparison between OpenABE and PALISADE-ABE.

Feature	OpenABE	PALISADE-ABE
Universe type	Large	Small
Attribute semantics	Multi-valued	Boolean (+1/−1)
Policy expressiveness	AND/OR + threshold	AND-only
Security model	CCA (classical)	s-CPA (post-quantum)
Primitive	Pairings (RELIC, BN curves)	Lattices (LWE, HEStd)
Implementation language	C++	C++

required and forbidden attributes. Nevertheless, the overall policy size in PALISADE tends to grow due to its exhaustive encoding.

– **Security proofs.** OpenABE offers Chosen-Ciphertext Attack (CCA) security in the classical setting, while PALISADE provides Selective Chosen-Plaintext Attack (sCPA) security under lattice assumptions. Notably, although CCA provides stronger guarantees, its proof in OpenABE relies on the DDH assumption, which is vulnerable to quantum attacks. PALISADE, based on the hardness of LWE, benefits from conjectured post-quantum resilience.

3.1 Plaintext

We encrypt messages of 256 bits, simulating a hybrid encryption scenario where the ABE scheme secures a symmetric key (*e.g.*, AES-256). This reflects a widely adopted usage model in real-world applications, where asymmetric ABE is used to protect symmetric encryption keys.

3.2 Security level

To ensure consistency, we adopt a 128 bit security level for both schemes.

In OpenABE many different curve families are available, including the well-known BN [55] and the BLS [56] classes. For our analysis, we opt for the BN382 curve provided by the RELIC library [54], rather than the default BN254, which recent cryptanalytic advances have shown to offer less than 100 bit security [57]. The BN382 curve is chosen being the strongest pairing-friendly curve readily available within the library that still offers a nominal 128 bit security level.

PALISADE supports multiple security levels based on the Homomorphic Encryption Security Standard (HEstd) [58]. We configure it with `HEStd128`, corresponding to 128 bit post-quantum security.

3.3 Attributes and policies

We evaluate performance across different attribute set sizes, namely

$$\mathcal{K} = \{6, 8, 16, 20, 32\}$$

which are the universe sizes for which parameters are estimated in PALISADE implementation when paired with `HEStd128`; exceeding 32 attributes makes the execution time grow unpractical [59].

In each configuration, the user is assigned all n attributes, and the corresponding access policy is designed to require all of them. This choice simulates a worst-case scenario, maximising the computational workload while maintaining compatibility across the two libraries.

The policy structure, though logically equivalent in both settings, is instantiated differently:

$$S = \{a_1, \ldots, a_n\}, \qquad W = a_1 \wedge \cdots \wedge a_n,$$

where a_i are the attributes. In OpenABE, this results in an access tree with one internal and gate and n leaves. In PALISADE, the policy is encoded as a vector of length n over $\{-1, 0, +1\}$, with all entries set to $+1$. Figure 2 illustrates this correspondence.

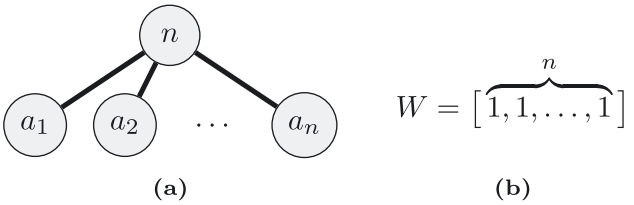

Figure 2: The same access structure devised in (a) OpenABE and (b) PALISADE.

3.4 Metrics

Each benchmark consists of 1,000 iterations for every attribute size $n \in \mathcal{K}$. For each run, we record the execution time (in milliseconds) of the four standard ABE operations: Setup, Key Generation, Encryption, and Decryption.

To assess performance, we compute the mean runtime of each operation across all trials, serving as a baseline indicator of average efficiency. To capture variability and potential edge behaviours, we also report classical statistical descriptors, including the standard deviation, selected quantiles, and the 1st percentile (*i.e.*, the 1 %-out). These metrics are presented via boxplots, which visually depict trends, dispersion, and outliers, providing an intuitive overview of how performance evolves with increasing attribute set sizes.

4 Results and discussion

We now present and discuss the benchmarking results, highlighting the relative strengths and limitations of the two schemes under evaluation. The analysis is structured around the four core operations of attribute-based encryption: Setup, Key Generation, Encryption, and Decryption.

Benchmarks are conducted on two distinct hardware platforms representing opposite ends of the performance spectrum. The high-end setup is an Ubuntu 20.04 Virtual Machine (VM) hosted on the CloudShare platform provided by the NTT Innovation Lab. It runs on a 16-core Intel Xeon E5-2660 v4 CPU at 2.00 GHz and has access to 400 GB of RAM. This environment mimics the computational capabilities of a CA or a dedicated cloud-based key management service, which can comfortably handle cryptographic workloads at scale.

In contrast, the low-end setup consists of a Raspberry Pi 4B (RPi), equipped with a quad-core ARM Cortex-A72 processor clocked at 1.8 GHz and 4 GB of LPDDR4-3200 SDRAM. This device emulates the resource constraints typical of edge computing environments, embedded systems, or lightweight clients participating in Internet of Things (IoT) infrastructures.

The rationale behind selecting such heterogeneous hardware lies in the desire to assess the scalability and adaptability of the two schemes across a broad spectrum of real-world scenarios. The VM provides a soft upper bound on achievable performance, representative of powerful centralised entities such as cloud servers or trusted authorities. Conversely, the RPi establishes a practical lower bound, capturing the limitations faced by constrained devices operating at the edge of the network: as an example, the RPi was not able to run the benchmark under the dockerised environment which had originally been designed for the task. By comparing results across both platforms, we can better understand the computational demands imposed by each scheme and the trade-offs involved when deploying them in settings with varying degrees of available resources.

In terms of raw performance impact (see also Figure 3(a)), the hardware differences manifest in a consistent and measurable way. Across both libraries, the RPi exhibits an approximately linear degradation in computational time when compared to the VM. The slowdown factor varies depending on the specific operation, ranging from about 1.5× (encryption in PALISADE) up to 4.5× (decryption in PALISADE). This behaviour confirms the predictable (yet substantial) influence that hardware capabilities have on execution times and underscores the importance of evaluating the cryptographic schemes on devices with markedly different performance profiles.

In Figure 3(b), we summarise the comparative analysis, reporting, for each function, the library exhibiting superior performance along with the associated average execution time ratio. In brief, OpenABE demonstrates a clear advantage during the SETUP and KEYGEN phases, whereas encryption performance is nearly equivalent across both implementations. In contrast, PALISADE shows a thorough advantage in the DECRYPT phase, significantly outperforming OpenABE (up to ∼73× in performance).

The remainder of this section presents a detailed breakdown of these findings across both hardware platforms, with a particular emphasis on the scalability of each scheme with respect to attribute universe size, as well as their adaptation to the two benchmarked computational environments.

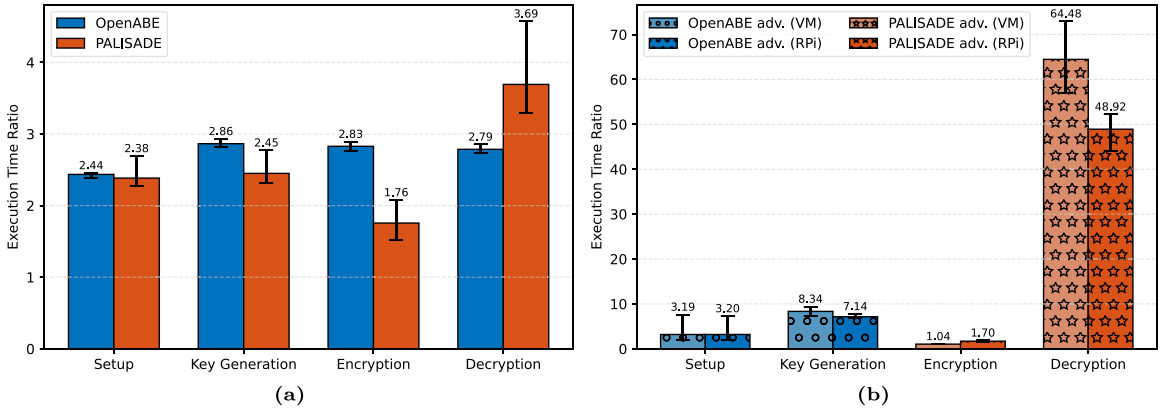

Figure 3: Summary of the performance gain (ratios) amongst the different hardware platforms and cryptographic schemes. Annotated bars indicate average speed-up ratios, while grey error bars represent the minimum and maximum observed values. (a) Relative advantage of executing each library on the VM with respect to the RPi. (b) Comparative advantage between OpenABE and PALISADE on both platforms: bars denote either an OpenABE advantage (blue, circles) or a PALISADE advantage (orange, stars), depending on which library performs better for a given operation.

4.1 Setup

Figure 4 presents the execution times for the `Setup` operation across increasing attribute sizes. As expected, OpenABE exhibits consistently constant timings, independent of the number of attributes. In contrast, PALISADE shows a clear linear growth, reaching a maximum difference of approximately 276 ms at 32 attributes on the high-end VM.

This observation aligns well with the theoretical foundations of the underlying schemes. OpenABE implements a large-universe construction, in which the setup phase is decoupled from the attribute set. Conversely,

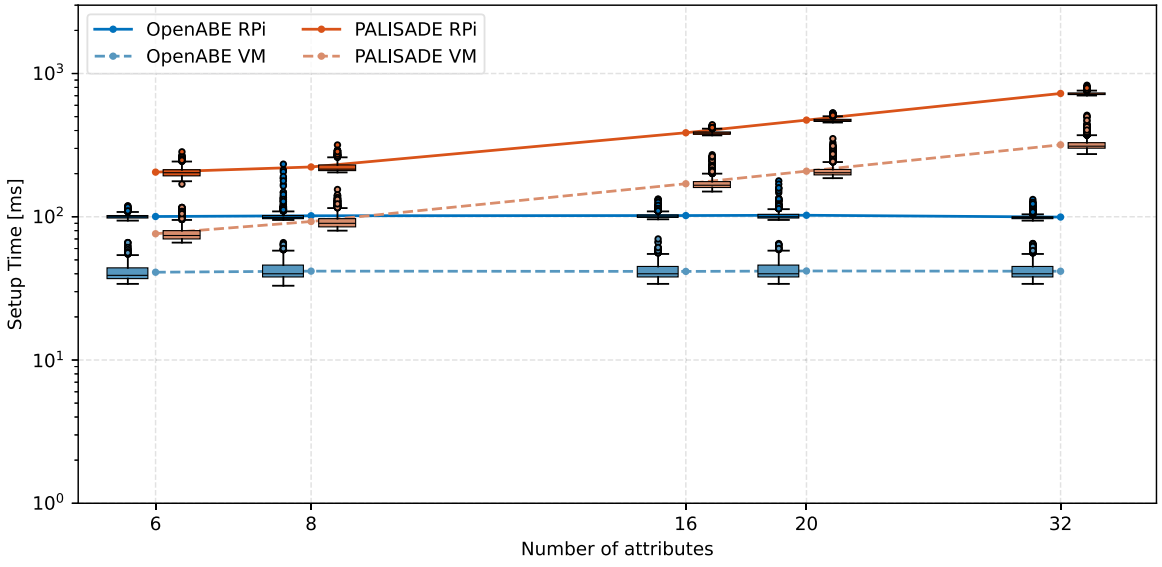

Figure 4: Execution times in milliseconds of the setup function. Results are obtained over 1,000 repetitions and summarised as boxplots (*x*-shifted for readability). Do notice that *x*-scale is logarithm (base-2) to preserve linearity w.r.t. the *y*-axis (log-10).

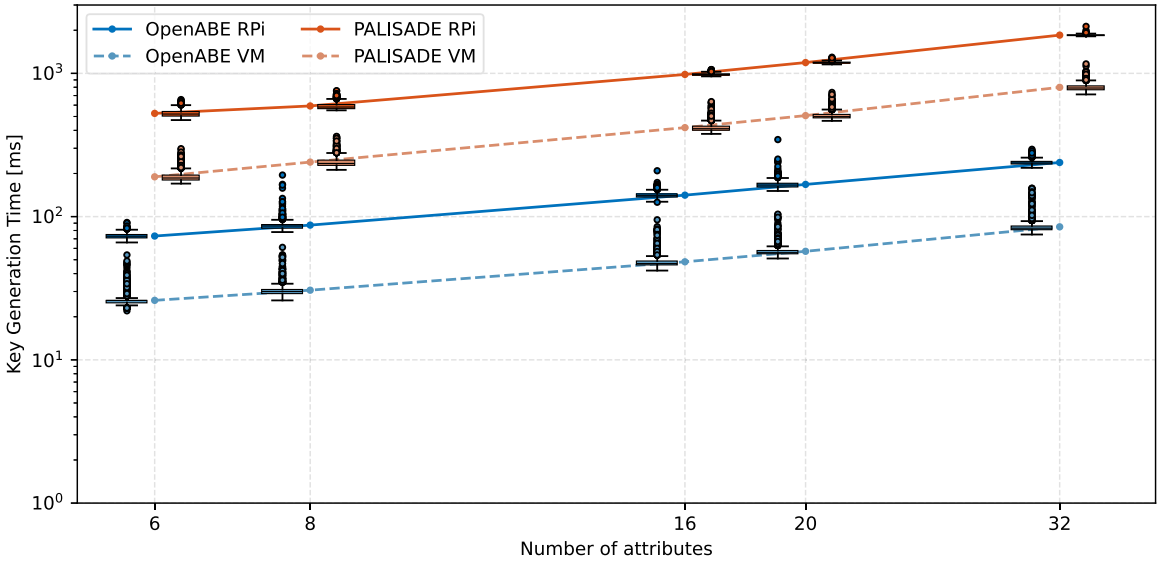

Figure 5: Execution times in milliseconds of the key generation function. Results are obtained over 1,000 repetitions and summarised as boxplots (*x*-shifted for readability). Do notice that *x*-scale is logarithm (base-2) to preserve linearity w.r.t. the *y*-axis (log-10).

the lattice-based scheme in PALISADE ties public parameter generation to each attribute, thus justifying the observed linear growth.

It is worth noting, however, that in most deployment scenarios, `Setup` is performed only once during the system's initialisation. As such, even a considerable difference in execution time may have limited practical impact.

When considering the performance on the, the same trend persists. The delta between the two schemes increases to roughly 714 ms for the maximum tested attribute set, again reinforcing the linear behaviour of PALISADE under resource constraints and the stable, hardware-independent cost of OpenABE's design.

4.2 Key generation

Figure 5 presents the execution times for the `Key Generation` operation across increasing attribute sizes.

Both schemes display a linear growth in execution time as the number of attributes increases. However, the performance gap between them is substantial: OpenABE consistently outperforms PALISADE, with speed-up factors ranging from approximately 7× to over 9× on the VM, and similarly between 6.8× and 7.7× on the RPi.

To provide a concrete comparison, for 32 attributes, OpenABE completes the operation in roughly 85 ms on the VM and 239 ms on the RPi. In contrast, PALISADE takes nearly 799 ms and 1.85 s, respectively – clearly demonstrating the computational weight of the lattice-based approach.

This difference is particularly relevant in practice. Unlike the Setup phase, which is executed only once during system initialisation, `Key Generation` must be run for each user enrolment. In large-scale deployments with many users, the high cost of PALISADE's key generation becomes a notable scalability concern.

That said, the computational burden may be mitigated in part by the deployment context. Since the system authority typically handles key generation, it is often equipped with sufficient computational resources to absorb the cost (the VM in our experiments). As such, although PALISADE imposes a heavier load, this may be acceptable in centralised or cloud-based environments, or when credentials are not constantly issued.

Additionally, PALISADE supports a division of the key generation procedure into offline and online phases, enabling further optimisation. However, we opted not to include this feature in the benchmarks as it is not implemented in the published version of the code and in order to maintain a fair comparison with OpenABE,

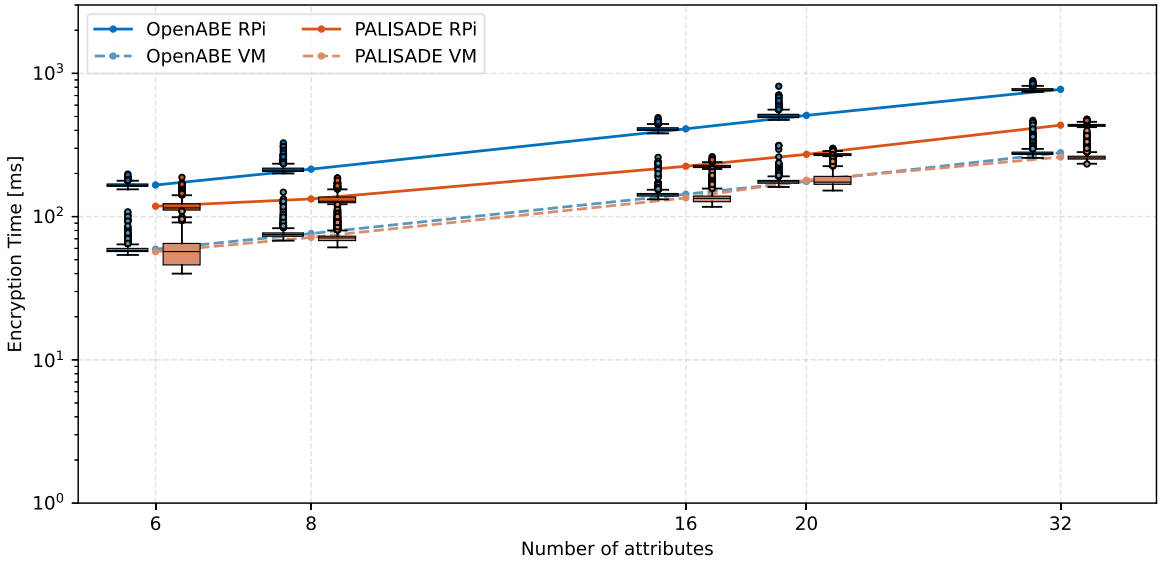

Figure 6: Execution times in milliseconds of the encryption function. Results are obtained over 1,000 repetitions and summarised as boxplots (*x*-shifted for readability). Do notice that *x*-scale is logarithm (base-2) to preserve linearity w.r.t. the *y*-axis (log-10).

which does not support an analogous optimisation. This also reflects our benchmarking principle of comparing default, standard implementations without relying on library-specific enhancements such as OpenABE's multivalued attributes or PALISADE's partial precomputation.

4.3 Encryption

Figure 6 presents the execution times for the `Encryption` operation.

Both schemes exhibit a linear increase in runtime as the number of attributes grows, and their performance remains closely aligned across the entire range. On the VM, execution times for both libraries converge around 270 ms for 32 attributes, with PALISADE slightly outperforming OpenABE by up to 6.9 % on smaller instances, and OpenABE regaining parity or a small edge at higher counts. On the RPi, however, OpenABE is notably slower, especially as the attribute count grows, with PALISADE completing encryption in roughly 56 %–71 % of the time required by OpenABE.

This gap on constrained devices is particularly relevant when considering that the encryption phase is typically executed by the data owner, who might not always have access to powerful hardware. In practical applications, this role may be fulfilled by mid-to low-end devices such as embedded systems or sensors deployed in IoT environments. The more efficient performance of PALISADE under such conditions suggests an advantage in scenarios where computational capabilities are limited, further reinforcing its suitability for edge-centric deployments. Performance remains essentially equivalent in high-end settings, instead.

One noteworthy distinction lies in the variability of execution times. The boxplot in Figure 6 shows that PALISADE experiences higher variance across 1,000 iterations, particularly for larger attribute sizes. This might be attributed to the underlying lattice-based computations, which may be more sensitive to system-level scheduling or internal randomness.

While these results suggest that encryption efficiency is not a bottleneck for either scheme, it is worth noting that PALISADE's current implementation limits the number of supported attributes to 32. This restricts our ability to assess scalability beyond that threshold, which may become relevant in high-complexity policy scenarios.

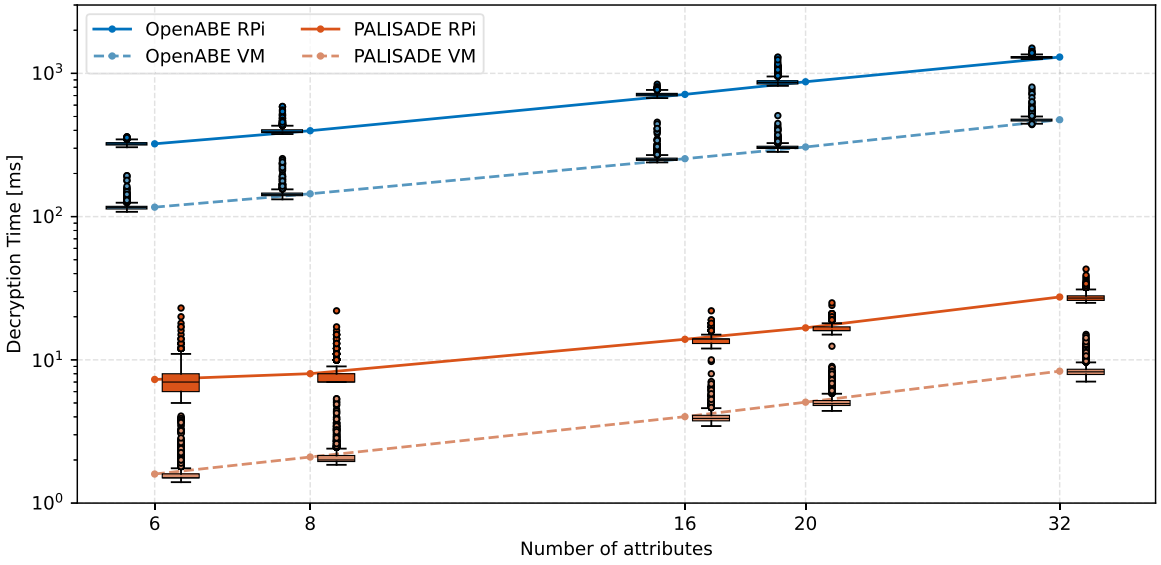

Figure 7: Execution times in milliseconds of the decryption function. Results are obtained over 1,000 repetitions and summarised as boxplots (*x*-shifted for readability). Do notice that *x*-scale is logarithm (base-2) to preserve linearity w.r.t. the *y*-axis (log-10).

4.4 Decryption

Finally, Figure 7 depicts the execution times of the `Decryption` function.

This is the only operation where PALISADE clearly and consistently outperforms OpenABE. While the pairing-based solution exhibits a substantial linear increase in execution time with the number of attributes – actually reaching up to 1.3 s on the – PALISADE maintains nearly constant decryption times across all attribute counts instead. The observed gap is considerable: PALISADE is approximately 57–73 times faster than OpenABE on the VM and 44 to 52 times faster on the RPi.

This behaviour is not surprising and stems directly from the underlying cryptographic constructions. In the lattice-based approach adopted by PALISADE, decryption primarily involves a matrix-vector multiplication followed by a rounding step to eliminate LWE noise (see Appendix A) These operations are lightweight and scale favourably. In contrast, OpenABE's pairing-based scheme relies on a number of pairing evaluations that grows with the policy size (see Appendix B). These pairings are computationally expensive and dominate the cost of decryption.

Despite PALISADE's more limited support for expressive access structures, the extremely low decryption cost makes it a strong candidate for deployment in constrained environments. This is especially important in practice, as decryption is typically performed by end-users, who are often the most resource-limited participants in the system. Moreover, in the context of attribute-based encryption, decryption tends to be the most frequently executed operation: multiple users may independently decrypt the same ciphertext, possibly many times over its lifetime.

Keeping decryption costs low is thus crucial for usability and responsiveness in real-world deployments. From this perspective, PALISADE offers a clear advantage, delivering efficient performance even on low-end devices.

5 Conclusions

In this work, we present the first comparative benchmark of CP-ABE schemes built upon pairing- and lattice-based cryptographic primitives. Our study aims to investigate the trade-off between these two paradigms, focusing on their performance, expressiveness, and potential for post-quantum security. To ensure a fair and

informative evaluation, we analyse two representative open-source C++ implementations: Waters' pairing-based scheme [12], implemented in OpenABE [11], and Zhang and Zhang's lattice-based scheme [4], implemented in PALISADE-ABE [9, 10]. Benchmarks are conducted over 1,000 repetitions per configuration and carried out on two hardware platforms: a high-end cloud-based virtual machine and a resource-constrained Raspberry Pi. Our analysis provides a comprehensive overview of how performances scale with the universe size and how the schemes adapt to different computational environments.

The results show that the lattice-based scheme exhibits remarkable decryption performance, consistently outperforming the pairing one in this phase, especially as the attribute size grows. This advantage holds across both hardware settings, highlighting its practical relevance. On the other hand, the pairing-based scheme maintains a slight advantage during the Setup and Key Generation phases, although the performance gap is modest and less relevant. Despite these results, a key limitation of lattice-based CP-ABE schemes remains the restricted policy expressiveness and attribute semantics. The pairing-based approach, though not post-quantum secure, currently provides far greater flexibility in terms of supported features and universe size.

From this analysis, it becomes evident that further research is needed to enhance the flexibility and feature set of lattice-based ABE schemes. Specifically, efforts should aim to introduce large-universe capabilities, richer policy constructions, and better integration of auxiliary functionalities such as key delegation.

Acknowledgments: Part of this work has been accepted for presentation at FCiR'25, the first international conference: Financial Cryptography in Rome. E. Onofri is a member of the "Gruppo Nazionale Calcolo Scientifico – Istituto Nazionale di Alta Matematica" (GNCS–INdAM). M. Pedicini is a member of the "Gruppo Nazionale per le Strutture Algebriche, Geometriche e le loro Applicazioni – Istituto Nazionale di Alta Matematica" (GNSAGA–INdAM).
Research ethics: Not applicable. The conducted research is not related to either human or animals use.
Informed consent: Not applicable.
Author contribution: All authors have accepted responsibility for the entire content of this manuscript and approved its submission. As common practice in cryptography, the authors have not followed the SDC approach; instead, they have sorted the authors alphabetically.
Use of Large Language Models, AI and Machine Learning Tools: Large language model–based AI tools were used solely to assist with language editing, including improving grammar, clarity, and the overall flow of the manuscript. These tools were not used for generating scientific content, data analysis, interpretation of results, or drawing conclusions. All authors reviewed, revised, and approved the final manuscript and take full responsibility for its content.
Conflict of interest: M. Pedicini is an editorial board member for *Journal and Mathematical Cryptology* and was not involved in the editorial review or the decision to publish this article. The authors declare there is no other conflict of interest.
Funding information: This work was partially supported by project SERICS (PE00000014) under the MUR National Recovery and Resilience Plan funded by the European Union - NextGenerationEU.
Data availability statement: Data sharing is not applicable to this article as no datasets were generated or analysed during the current study.
Software availability: The full codebase required to reproduce the experiments, together with the raw benchmark data used in this study, is publicly available under open-source license at: https://github.com/eOnofri04/Cross-primitive-comparison-in-CP-ABE.

Appendix A Bilinear pairing schemes

In this appendix, we cover the essentials of pairing-based ABE, highlighting its functionalities and main components.

Let $\mathbb{G}$, $\mathbb{H}$, and $\mathbb{G}_T$ be three multiplicative cyclic groups of prime order p and let g and h be two generators of $\mathbb{G}$ and $\mathbb{H}$ respectively. We recall the founding concept of bilinear pairing as a map

$$e : \mathbb{G} \times \mathbb{H} \to \mathbb{G}_T$$

such that the following three properties hold:
Bilinearity $e(g^a, h^b) = e(g, h)^{ab}$ for all $a, b \in \mathbb{Z}_p^\star$.
Non-degeneracy $e(g, h)$ is not the identity in $\mathbb{G}_T$.
Efficiency e is efficiently evaluable.

Pairings offer the main cryptographic primitive where all the operations of the scheme are embedded. Consequently, since we are interested in granting a sufficient level of (cryptographic) security, in what follows, we consider groups for which the DDH problem is assumed to be hard. A typical choice for source groups is, hence, considering elliptic curves as well as relying on the symmetric setting, *i.e.* where $\mathbb{G} = \mathbb{H}$, simplifying the discussion.

Originally introduced by Shamir in [60], let us also recall the notion of Secret-Sharing Scheme (SSS) as the main cryptographic primitive behind the typical access structure construction. In general, a SSS allows dividing a secret into multiple pieces (or shares, in the jargon) which can, in turn, be distributed amongst the parties of a system, possibly with some redundancy; when enough shares are collected, it is then possible to reconstruct the secret. Different ways of combining the shares to reconstruct the secret exist: in particular, the most widely adopted approach in pairing-based ABE constructions relies on linear mapping (a linear combination of the shares is evaluated), hence forming the so-called Linear Secret-Sharing Scheme (LSSS, we refer to [61] for a detailed description).

In the context of pairing-based ABE, policies are represented as boolean formulas on a set of ℓ attributes; a formula is then converted into a ℓ-rows matrix A with a proper procedure. A is used to generate a set of ℓ shares $\lambda = (\lambda_1, \dots, \lambda_\ell)$ whose reconstruction yields a ciphertext-related secret s to be employed in message encryption. Here, the relation $A \times v = \lambda$ holds, with v being a random vector containing s as the first entry; hence v can be retrieved as some linear combination of λ. The intended recipient can then reconstruct a matrix A' based on the same attribute set in order to retrieve the proper linear combination of λ to obtain s and hence decrypt the message.

A.1 The Waters' scheme

We can now proceed to briefly describe the pairing-based CP-ABE scheme introduced by Waters in [12] we have chosen for our benchmark experiment. The algorithm is based on a pairing e of $\mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$, with $\mathbb{G}$ being of prime order p, and exploits the functionality of the LSSS to implement the policy embedded in the ciphertext (see also [61]).

Given two public exponentiations $e(g, g)^\alpha$ and g^a, the encryption is achieved as the couple

$$M\ (e(g, g)^\alpha)^s, \qquad g^s,$$

where s is a cyphertext-related LSSS secret and α is a public exponent, along with the set of shares λ_i, hidden through a random secret r_i as

$$(g^a)^{\lambda_i} \cdot \xi^{-r_i}, \qquad g^{r_i}, \qquad i \in \{1, \dots, \ell\},$$

and ξ_i is a properly derived value obtained by the i-th attribute.

During the decryption phase, if the recipient is allowed to decrypt, *i.e.* she possesses a set of attributes S compliant with the policy, then she can obtain $e(g, g)^{\alpha s}$ by applying a series of pairings between the ciphertext components and the portion of her key associated with the proper attributes. The original message M can then be retrieved with the proper division operation $C / e(g, g)^{\alpha s}$ in $\mathbb{G}_T$, where the security is enforced by the fact that s can only be (virtually) retrieved from the correct (proper attribute) weighted combination of λ_i.

In particular, the large universe property of the Waters scheme is achieved using a proper hash function $H\colon \{0,1\}^* \to \mathbb{G}$ to embed any arbitrary string in a group element. This avoids defining *a priori* a number of elements equal to the number of attributes in the universe U and significantly reduces the size of the master public key.

Appendix B Lattice-based dual-Regev schemes

Within this appendix, we explore the main characteristics of lattice-based ABE schemes, providing the related details for the interested reader.

We recall a n-dimensional real lattice of rank $m \leq n$ being a subset of $\mathbb{R}^n$ given by the span of m linearly independent vectors $b_1, \ldots, b_m \in \mathbb{R}^n$, *i.e.*

$$\Lambda = \mathcal{L}(B) = \{\langle B, c\rangle | c \in \mathbb{Z}^m\},$$

where $B \in \mathbb{R}^{n\times m} = [b_1 \| \ldots \| b_m]$ is called *basis* of the lattice. Typical hard problems on lattices require finding minimal vectors with some properties on a properly built hard basis, see, *e.g.*, SIVP requiring finding n-independent "short" vectors. Their usage in cryptography is ensured by the presence of coupled short basis where the same solution can be easily found, see Algorithm 1 from [62] for the generation of such *Trapdoors*, *i.e.*`trapGen` algorithm.

In particular, we focus on the constructions based on the Regev Scheme [41], as this is later used in the scheme by Zhang and Zhang. We recall the Regev encryption scheme being the first lattice-based encryption scheme based on the hardness of the LWE problem, which is considered to be resistant to quantum attacks by the worst-case complexity of gapSVP and SIVP on lattices.

In Regev schemes (see LWE-PKE in [63]), messages are represented as vectors m and encryption is achieved by adding to a magnified version of the message ($\Delta \cdot m$) a secret key s (masked by a random linear combination) and a noise e sampled from a known distribution χ, *i.e.*:

$$c = \Delta m + \langle s, a\rangle + e, \qquad e \leftarrow_{\$} \chi, \qquad a \leftarrow_{\$} \text{Unif}.$$

Cyphertexts are hence obtained as couples (c, a), whose security is based on the indistinguishability from randomness obtained via the error e. Finally, decryption can only be efficiently done by the intended recipient who possesses the secret key by applying a rounding on $\Delta m + e$; noise, in fact, prevents recovering the secret key under known plaintext attacks, and message retrieval is possible as long as the noise is kept below the magnification factor Δ. In this context, the Gaussian distribution represents a typical choice for χ, as efficient sampling algorithms (see Theorem 4.1 from [63] for the `sampleGaussian` algorithm) and nice distribution properties (see Lemma 4.4 from [64]) were studied on real lattices.

Dual-Regev CP-ABE schemes can then be built on top of the Regev scheme by associating ciphertexts with access policies represented as pattern strings, where symbols can be 0, 1, or *. Users possess secret keys corresponding to their attributes (represented as bit strings). Decryption succeeds as usual if the user's attribute matches the policy pattern specified in the ciphertext.

We refer to our previous work [44] for any further details on the lattice context and how it is exploited for dual-Regev CP-ABE schemes. Furthermore, we also refer to the recent book [65] for a broader survey of the lattice-based cryptography literature.

B.1 The Zhang and Zhang's scheme

In what follows, we describe the scheme by Zhang and Zhang [4] implemented in PALISADE-ABE that we use in our benchmark. The scheme, based again on SSS techniques, uses a randomly chosen cyphertext-related shared secret $s \leftarrow_{\$} \mathbb{Z}_q^n$ to encrypt the message in a LWE-PKE fashion. The idea is that users can redeem the share if and

only if they have enough matching attributes to *interpolate* it, with the number and the type described in the access policy.

The scheme leverages two public matrices B_i^+ and B_i^- per attribute i, representing the *interpolation* coordinate of, respectively, the positive and the negative instance of i-th attribute. Apart from the public key, accounting also for the hard basis of the lattice, and a random vector $u \in \mathbb{Z}_q^n$, to be used in pair with the coordinate of the shares B_0, the cryptosystem also presents a master secret key holding the short basis, computed via `trapGen`.

The actual *interpolation* is achieved by hiding the secret in multiple LWE samples (one per each admissible attribute). The recipient is able to decrypt if its key is capable of inverting enough samples to retrieve the secret. Here, the inversion property is granted to the user key at its creation, thanks to the usage of the master secret key (*i.e.* of the short basis of the lattice) via the algorithm `genSamplePre` (see Theorem 3.4 from [66] for details).

References

1. Bethencourt J, Sahai A, Waters B. Ciphertext-Policy Attribute-Based Encryption. In: 2007 IEEE symposium on security and privacy (SP '07). IEEE; 2007:321—34 pp.
2. Mosteiro-Sanchez A, Barcelo M, Astorga J, Urbieta A. All cryptolibraries are beautiful, but some are more beautiful than others: a survey of CP-ABE libraries. In: 3rd URSI Atlantic/Asia-Pacific radio science meeting — 2022. URSI; 2022:1—4 pp.
3. Sandhia GK, Raja SVK. Secure sharing of data in cloud using MA-CPABE with elliptic curve cryptography. JAIHC 2021;13:3893—902.
4. Zhang J, Zhang Z. A ciphertext policy attribute-based encryption scheme without pairings. In: International conference on information security and cryptology. Springer; 2011:324—40 pp.
5. Boneh D. The decision Diffie—Hellman problem. In: Int. algorithmic number theory symposium. Springer; 1998:48—63 pp.
6. Diffie W, Hellman ME. New directions in cryptography. IEEE TIT 1976;22:644—54.
7. Shor PW. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. SIAM Rev 1999;41:303—32.
8. Peikert C. A decade of lattice cryptography. Found Trends® Theor Comput Sci 2016;10:283—424.
9. Rohloff K, Polyakov Y, Cousins D. OpenSource, editor. PALISADE-ABE. GitLab; 2022. Available from: https://gitlab.com/palisade/palisade-abe.
10. Rohloff K, Polyakov Y, Cousins D, Ryan G. OpenSource, editor. PALISADE cryptography library. GitLab; 2023. https://palisade-crypto.org [Accessed 10 Apr 2023].
11. Zeutro. OpenSource, editor. Zeutro/openabe: the openabe library — open source cryptographic library with attribute-based encryption implementations in C/C++. GitLab; 2018. Available from: https://github.com/zeutro/openabe.
12. Waters B. Ciphertext-Policy Attribute-Based Encryption: an expressive, efficient, and provably secure realization. In: Int. workshop on PKC. Springer; 2011:53—70 pp.
13. Akinyele JA, Garman C, Miers I, Pagano MW, Rushanan M, Green M, et al. Charm: a framework for rapidly prototyping cryptosystems. J Cryptogr Eng 2013;3:111—28.
14. Fraunhofer AISEC. OpenSource, editor. Rabe: rust attribute based encryption library. GitHub; 2023. Available from: https://github.com/Fraunhofer-AISEC/rabe.
15. Fentech Project. OpenSource, editor. CiFer. GitHub; 2021. Available from: https://github.com/fentec-project/CiFEr.
16. Fentech Project. OpenSource, editor. GoFe. GitHub; 2022. Available from: https://github.com/fentec-project/gofe.
17. Li J, Zhang Y, Ning J, Huang X, Poh GS, Wang D. Attribute based encryption with privacy protection and accountability for CloudIoT. IEEE TCC 2020;10:762—73.
18. Rouselakis Y, Waters B. Efficient statically-secure large-universe multi-authority attribute-based encryption. In: International conference on financial cryptography and data security. Springer; 2015:315—32 pp.
19. Yamada S, Attrapadung N, Hanaoka G, Kunihiro N. A framework and compact constructions for non-monotonic attribute-based encryption. In: Public-key cryptography — PKC 2014. Springer; 2014:275—92 pp.
20. Chen J, Gay R, Wee H. Improved dual systemABE in prime-order groups via predicate encodings. In: Annual international conference on the theory and applications of cryptographic techniques. Springer; 2015:595—624 pp.
21. Agrawal S, Chase M. FAME: fast attribute-based message encryption. In: Proceedings of the 2017 ACM SIGSAC conference on computer and communications security; 2017:665—82 pp.
22. Li J, Yao W, Han J, Zhang Y, Shen J. User collusion avoidance CP-ABE with efficient attribute revocation for cloud storage. IEEE Syst J 2017;12:1767—77.
23. Liu Q, Wang G, Wu J. Time-based proxy re-encryption scheme for secure data sharing in a cloud environment. Inf Sci 2014;258:355—70.
24. Yang Y, Liu JK, Liang K, Choo KKR, Zhou J. Extended proxy-assisted approach: achieving revocable fine-grained encryption of cloud data. In: Computer security — ESORICS 2015. Springer; 2015:146—66 pp.

25. Lewko A, Waters B. Decentralizing attribute-based encryption. In: Annual international conference on the theory and applications of cryptographic techniques. Springer; 2011:568—88 pp.
26. Bramm G, Gall M, Schütte J. DBABE: blockchain-based distributed attribute based encryption. In: 15th international joint conference on e-business and telecommunications (ICETE 2018); 2018, vol 2:99—110 pp.
27. Sahai A, Waters B. Fuzzy identity-based encryption. In: Advances in cryptology — EUROCRYPT 2005. Springer; 2005:457—73 pp.
28. Shamir A. Identity-based cryptosystems and signature schemes. In: Lecture notes in computer science. Berlin, Heidelberg: Springer; 2000:47—53 pp.
29. Moffat S, Hammoudeh M, Hegarty R. A survey on Ciphertext-Policy Attribute-Based Encryption (CP-ABE) approaches to data security on mobile devices and its application to IoT. In: Proceedings of ICFNDS17: Assoc. Computing Machinery. New York, NY, USA; 2017:1—10 pp.
30. Rasori M, Manna ML, Perazzo P, Dini G. A survey on attribute-based encryption schemes suitable for the internet of things. IEEE Internet Things J 2022;9:8269—90.
31. Zhang Y, Deng RH, Xu S, Sun J, Li Q, Zheng D. Attribute-based encryption for cloud computing access control: a survey. ACM Comput Surv 2020;53:1—41.
32. Goyal V, Pandey O, Sahai A, Waters B. Attribute-based encryption for fine-grained access control of encrypted data. In: Proceedings of the 13th ACM conference on computer and communications security. CCS06. ACM; 2006:89—98 pp.
33. Sahai A, Okamoto T, Takashima K, Waters B. Fully secure functional encryption: attribute-based encryption and (hierarchical) inner product encryption. In: Advances in cryptology — EUROCRYPT 2010. Springer; 2010:62—91 pp.
34. Attrapadung N. Dual system encryption framework in prime-order groups via computational pair encodings. In: Advances in cryptology — ASIACRYPT 2016. Springer; 2016:280—311 pp.
35. Waters B. Dual system encryption: realizing fully secure IBE and HIBE under simple assumptions. In: Annual international cryptology conference. Springer; 2009:619—36 pp.
36. Venema M, Alpár G, Hoepman JH. Systematizing core properties of pairing-based attribute-based encryption to uncover remaining challenges in enforcing access control in practice. Des Codes Cryptogr 2023;91:165—220.
37. de la Piedra A, Venema M, Alpár G. ABE squared: accurately benchmarking efficiency of attribute-based encryption. Cryptology ePrint Archive 2022;2022. https://eprint.iacr.org/2022/038.
38. Mosteiro-Sanchez A, Barcelo M, Astorga J, Urbieta A. Too many options: a survey of ABE libraries for developers. arXiv 2022. https://arxiv.org/abs/2209.12742.
39. Chen L, Jordan S, Liu YK, Moody D, Peralta R, Perlner R, et al. Report on post-quantum cryptography. Washington D.C.: National Institute of Standards and Technology; 2016.
40. Nejatollahi H, Dutt N, Ray S, Regazzoni F, Banerjee I, Cammarota R. Post-quantum lattice-based cryptography implementations: a survey. ACM Comput Surv (CSUR) 2019;52:1—35.
41. Regev O. On lattices, Learning With Errors, random linear codes, and cryptography. In: Proceedings of ACM STOC'05. New York: ACM; 2005:84—93 pp.
42. Zhang J, Zhang Z, Ge A. Ciphertext policy attribute-based encryption from lattices. In: Proceedings of the 7th ACM symposium on information, computer and communications security; 2012:16—7 pp.
43. Chen Z, Zhang P, Zhang F, Huang J. Ciphertext policy attribute-based encryption supporting unbounded attribute space from R-LWE. KSII — TIIS 2017;11:2292—309.
44. Cianfriglia M, Onofri E, Pedicini M. mR_{LWE}-CP-ABE: a revocable CP-ABE for post-quantum cryptography. J Math Cryptol 2024;18:20230026.
45. Datta P, Komargodski I, Waters B. Decentralized multi-authorityABE for DNFs from LWE. In: Advances in cryptology — EUROCRYPT 2021. Springer International Publishing; 2021:177—209 pp.
46. Yao YF, Chen HY, Gao Y, Wang K, Yu HY. A decentralized multi-authority CP-ABE scheme from LWE. J Inf Secur Appl 2024;82:103752.
47. Ma C, Gao H, Hu B. Ciphertext policy attribute-based encryption scheme supporting Boolean circuits over ideal lattices. J Inf Secur Appl 2024;84:103822.
48. Wee H. Optimal broadcast encryption and CP-ABE from evasive lattice assumptions. In: Dunkelman O, Dziembowski S, editors. Advances in cryptology — EUROCRYPT 2022. Cham: Springer International Publishing; 2022:217—41 pp.
49. Waters B, Wee H, Wu DJ. Multi-authority ABE from lattices without random oracles. In: Kiltz E, Vaikuntanathan V, editors. Theory of cryptography. Cham, Switzerland: Springer Nature; 2022:651—79 pp.
50. Hsieh YC, Lin H, Luo J. A general framework for lattice-based ABE using evasive inner-product functional encryption. In: Joye M, Leander G, editors. Advances in cryptology — EUROCRYPT 2024. Cham, Switzerland: Springer Nature; 2024:433—64 pp.
51. Wee H. Almost optimal KP and CP-ABE for circuits from succinct LWE. In: Fehr S, Fouque PA, editors. Advances in cryptology — EUROCRYPT 2025. Cham, Switzerland: Springer Nature; 2025:34—62 pp.
52. Zhang R, Li J, Lu Y, Han J, Zhang Y. Key escrow-free attribute based encryption with user revocation. Inf Sci 2022;600:59—72.
53. Das S, Namasudra S. Multiauthority CP-ABE-based access control model for IoT-enabled healthcare infrastructure. IEEE TII 2023;19:821—9.
54. RELIC Toolkit Contributors. OpenSource, editor. RELIC toolkit: a modern cryptographic library. GitHub; 2023. Available from: https://github.com/relic-toolkit/relic.

55. Barreto PS, Naehrig M. Pairing-friendly elliptic curves of prime order. In: International workshop on selected areas in cryptography. Springer; 2005:319—31 pp.
56. Barreto PS, Lynn B, Scott M. Constructing elliptic curves with prescribed embedding degrees. In: Security in communication networks: third international conference, SCN 2002 Amalfi, Italy, september 11—13, 2002 revised papers 3. Springer; 2003:257—67 pp.
57. Sakemi Y, Kobayashi T, Saito T, Wahby RS. Pairing-friendly curves. Internet engineering task force. draft-irtf-cfrg-pairing-friendly-curves-11. Work in Progress; 2022. Available from: https://datatracker.ietf.org/doc/draft-irtf-cfrg-pairing-friendly-curves/11/.
58. Albrecht M, Chase M, Chen H, Ding J, Goldwasser S, Gorbunov S, et al. Homomorphic encryption security standard. Toronto, Canada: HomomorphicEncryption.org; 2018.
59. Abdennebi A, Savaş E. Archive Ce, editor. A lattice-based publish-subscribe communication protocol using accelerated homomorphic encryption primitives. IACR; 2023. Available from: https://eprint.iacr.org/2023/1309.
60. Shamir A. How to share a secret. Commun ACM 1979;22:612—3.
61. Beimel A, Chor B, editors. Secure schemes for secret sharing and key distribution. Israel Institute of Technology — Haifa; 1996. Available from: https://www.cs.bgu.ac.il/beimel/Papers/thesis.pdf.
62. Alwen J, Peikert C. Generating shorter bases for hard random lattices. Theor Comput Syst 2010;48:535—53.
63. Gentry C, Peikert C, Vaikuntanathan V. Trapdoors for hard lattices and new cryptographic constructions. In: 40th annual ACM symposium on theory of computing; 2008:197—206 pp.
64. Micciancio D, Regev O. Worst-case to average-case reductions based on Gaussian measures. SIAM J Comput 2007;37:267—302.
65. Zhang J, Zhang Z. Lattice-based cryptosystems: a design perspective. Singapore: Springer; 2020.
66. Cash D, Hofheinz D, Kiltz E, Peikert C. Bonsai trees, or how to delegate a lattice basis. In: Gilbert H, editor. Advances in cryptology — EUROCRYPT 2010. Berlin, Heidelberg: Springer; 2010:523—52 pp.