PICADOR: End-to-End Encrypted Publish-Subscribe Information Distribution with Proxy Re-Encryption[☆]

Cristian Borcea[a], Arnab "Bobby" Deb Gupta[a], Yuriy Polyakov[a,b], Kurt Rohloff[a,*], Gerard Ryan[a]

[a] *NJIT Cybersecurity Research Center and Dept. of Computer Science New Jersey Institute of Technology, Newark, NJ 07102, USA*
[b] *CSAIL, Massachusetts Institute of Technology, Cambridge, MA 02139, USA*

Abstract

This article presents PICADOR, a system for end-to-end encrypted **P**ubl**i**sh-Subs**c**ribe inform**a**tion **d**istribution with pr**o**xy **r**e-encryption. PICADOR is designed for topic-based Pub/Sub systems and provides end-to-end payload confidentiality. The main novelty of PICADOR is that it provides an information distribution service with end-to-end encryption where publishers and subscribers do not need to establish shared encryption and decryption keys. Multiple publishers post encrypted information to a Pub/Sub broker which uses Proxy Re-Encryption (PRE) to convert this information into a representation that can only be decrypted by approved subscribers. The broker is unable to decrypt the information. To support PICADOR, we design and implement a novel PRE scheme that leverages a general lattice encryption software library. We prototype our system using a scalable Java-based information substrate that supports topic-based Pub/Sub operations. We experimentally evaluate performance and scalability tradeoffs in the context of enterprise and mobile applications. We discuss design tradeoffs and application-specific customizations.

Keywords: security, encryption, information brokering
2010 MSC: 94A60, 68P25, 92C50

1. Introduction

Publish-Subscribe (Pub/Sub) information distribution provides a flexible and asynchronous approach to distribute information between information pro-

[☆]Project sponsored by the National Security Agency under Grant H98230-15-1-0274 and the National Science Foundation (NSF) under Grants No. CNS 1409523 and DGE 1565478. The United States Government is authorized to reproduce and distribute reprints notwithstanding any copyright notice herein.

[*]Corresponding author
Email addresses: `borcea@njit.edu` (Cristian Borcea), `ad479@njit.edu` (Arnab "Bobby" Deb Gupta), `polyakov@njit.edu` (Yuriy Polyakov), `rohloff@njit.edu` (Kurt Rohloff), `gwryan@njit.edu` (Gerard Ryan)

ducers (i.e., publishers) and information consumers (i.e., subscribers) [1]. This distribution model is useful in environments where information needs to be aggregated from possibly multiple sources and distributed to consumers that have no direct connections with the information producers. Publishers and subscribers do not need to interact with each other directly or share identity information. There are two main types of Pub/Sub systems: topic-based and content-based. This article focuses on topic-based Pub/Sub systems: publishers post information with metadata topic labels to a Pub/Sub broker, and this broker distributes the published information to subscribers that are registered for specific topics.

An example use case that benefits from Pub/Sub systems is in an enterprise medical domain where patients' past medical records need to be shared with emergency care providers. In this scenario an insurance provider can operate a Pub/Sub system to support the distribution of past medical records [2, 3]. Another example, in a tactical/mobile military domain, is when multiple military units passing through a geographic region generate imagery or other information. The units may be unable to interact directly [4, 5], but there would be clear benefits from sharing information. In this military example, a local Forward Operating Base or an Unmanned Aerial Vehicle flying overhead could maintain a Pub/Sub broker to support information sharing between units.

A major challenge for Pub/Sub systems, illustrated by the application domains of these two examples, is confidentiality of information which is distributed by the Pub/Sub broker. Existing Pub/Sub systems protect information payloads via encryption that requires either: 1) the publisher and subscriber coordinate to establish the encryption and decryption keys or 2) the Pub/Sub broker decrypts the information payloads from the publishers and then encrypts this information payload again for re-transmission to the subscribers. The first solution contradicts one of the goals of Pub/Sub systems, i.e., the decoupling of publishers and subscribers. The second solution solves this issue, but gives the broker access to the unprotected information. Thus, it makes the broker a ripe target for adversaries to compromise and steal sensitive information [6].

These security limitations could be mitigated by running the Pub/Sub broker only in trusted computing environments, but these environments can be expensive to set up and maintain because they require specialized management and housing in dedicated facilities [7]. Further, restriction to deployment in trusted environments limits the practical applicability of Pub/Sub systems.

This article presents PICADOR, a system for end-to-end encrypted **P**ubl**i**sh-Subs**c**ribe inform**a**tion **d**istribution with pr**o**xy **r**e-encryption. PICADOR uses Proxy Re-Encryption (PRE) [8, 9] to maintain end-to-end encryption of payload from publishers to subscribers in topic-based Pub/Sub systems. Our system does not require publishers or subscribers to interact directly a priori. PICADOR's PRE capability enables secure re-encryption of the publisher-encrypted payload at the broker such that the payload can be decrypted by authorized subscribers. PICADOR guarantees that the broker cannot decrypt payload information. In this way, PICADOR ensures end-to-end confidentiality, while improving the decoupling of publishers and subscribers.

In addition to the PICADOR system architecture, the second major contribution of this article is a novel PRE scheme, based on a provably secure lattice-based cryptosystem [10]. This PRE scheme is unidirectional. Our encryption design has also the benefit of being "post-quantum", meaning that it is resistant to quantum computing attacks [11, 12, 13]. The PRE techniques in PICADOR could also be combined with subscription privacy techniques [14] to protect the privacy of subscriptions (i.e., topic metadata, which is maintained unencrypted to enable brokering) in addition to the confidentiality of the information payloads.

We prototype PICADOR using a scalable Java-based information substrate that supports the Pub/Sub operations. Experimental results demonstrate that PICADOR's performance is practical in application domains such as healthcare and military. PICADOR provides low latency for text-based applications, and it works for multimedia applications that do not have stringent latency constraints. We experimentally observe that PICADOR operations are compute-bound, meaning that performance can be improved with relatively simple and cost-effective multi-threading and multi-core enhancements rather than more expensive and often infeasible increased bandwidth provisioning.

The rest of the article is organized as follows. Section 2 presents an overview of PICADOR. Section 3 describes our novel PRE scheme and analyzes its security. Section 4 covers the architecture and implementation of PICADOR. Section 5 evaluates our system and presents insights derived from our experimental results. Section 6 presents related work. Section 7 discusses conclusions and future work.

2. PICADOR Overview

PICADOR has two main contributions: (1) a novel system architecture for providing payload confidentiality in topic-based Pub/Sub systems; and (2) a new PRE scheme that enables our cryptographic protocols for secure publisher/subscriber interaction. The system architecture solves the challenge of designing secure Pub/Sub interaction mechanisms that better decouple direct publisher/subscriber interactions for key management while maintaining payload confidentiality. The new PRE scheme solves the practicality/performance challenges encountered by public-key PRE approaches and increases the decoupling of publisher/subscriber interactions.

PICADOR's system architecture uses Proxy Re-Encryption (PRE) capabilities to allow subscribers to receive and decrypt encrypted data that they are intended to receive without ever directly coordinating their keys with the publisher of the data. PRE [15] enables secure re-encryption of ciphertexts from one secret key to another without decryption. From a high level point of view, publishers tag their content with topical metadata and encrypt the content data. The Pub/Sub broker receives data encrypted by the publisher and re-encrypts it such that only the intended subscribers can decrypt it. Possession of the re-encryption key does not provide the broker with access to the data. In this way, end-to-end confidentiality of the encrypted payload data is guaranteed between subscribers and publishers. Furthermore, the PRE-enabled broker allows for flexible communication with topic-based brokering, as new publishers, subscribers and topics can be added at any time to the system. The broker can re-encrypt data for these new topics and subscribers even though the publishers or the broker were not aware of them a priori.

Our novel PRE scheme solves two main practical challenges regarding ciphertext expansion and operation latency. The challenge of ciphertext expansion is that a ciphertext may be much larger than plaintext. The problem of latency is that encryption, decryption, and re-encryption operations may take too long for applications that require low latency. Our PRE scheme provides a low ciphertext expansion, meaning that PICADOR is space efficient, uses less disk space to encrypt information, reduces bandwidth usage requirements, and overall leads to practical throughput. The scheme is also time efficient, resulting in low-latency communication in PICADOR as compared to other PRE scheme prototypes.

An admitted limitation of PICADOR is that it requires more computational and organizational overhead to register subscriptions, meaning that it may not be suitable in situations where many publishers and subscribers are added quickly after deployment. As mentioned Section 1, PICADOR also does not protect the privacy of registered subscriptions, with topic matching occurring in the clear, but this problem can be solved leveraging existing solutions [14]. The current design of PICADOR is also centralized, with a single broker instance. We discuss an approach to scale PICADOR with multiple brokers operating in parallel.

2.1. Threat Model

From the Confidentiality-Integrity-Assurance (CIA) security analysis perspective, PICADOR addresses confidentiality concerns of honest-but-curious adversaries which fully compromise the Pub/Sub broker. We do not focus on integrity and authentication issues as this is not directly addressed by our PRE-enabled contribution and focus on our confidentiality contributions. We can address availability (via replication) from a design perspective, but we primarily leave this to future work. We provide a more detailed discussion of integrity, authentication, and availability aspects in Section 7.

2.2. Basic Operation

The basic usage of PICADOR is shown in Figure 1. Alice wants to publish information to the PRE-enabled broker (i.e., Pub/Sub broker[1]), and Bob is the subscriber who is interested in this information. The figure also shows the clients that execute the publisher and the subscriber software. A Policy

[1]We use the term “broker” to mean the Pub/Sub and PRE software that is hosted on a “server”, and we use “server” to exclusively mean the computing device(s) that the Pub/Sub and PRE “broker” is hosted on.

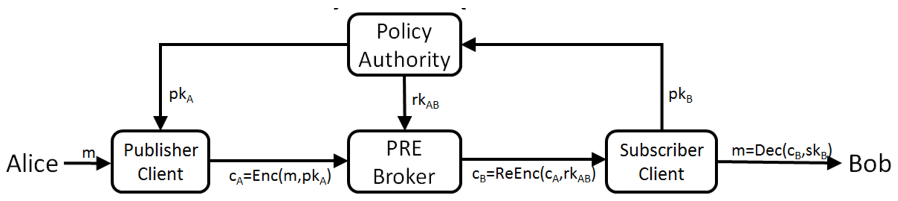

Figure 1: High Level PICADOR Operation

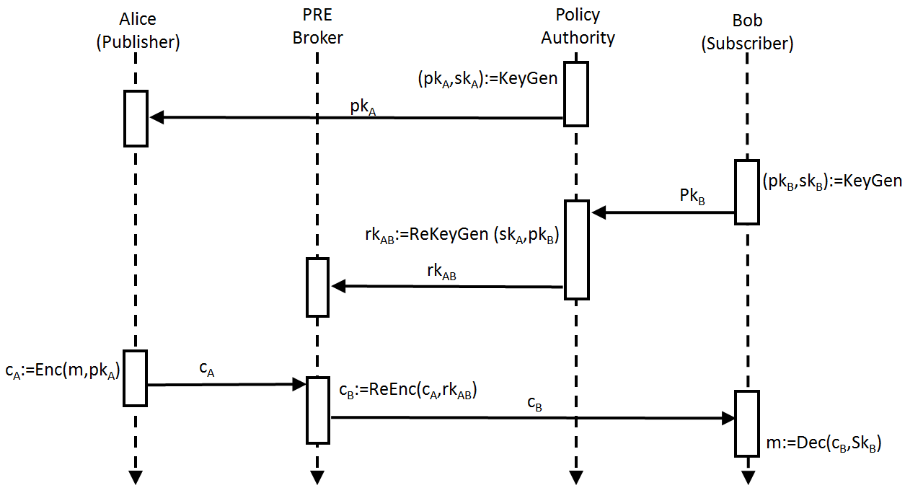

Figure 2: PRE Key Management and Interaction Workflow

Authority helps with key management and maintains control over who receives which information from the broker. We assume that the Policy Authority is trusted (e.g., it does not collude with a potentially honest-but-curios broker). The Policy Authority (PA) is intended to set re-encryption policy and not be involved in resource-intensive operations such as data encryption or proxy re-encryption.

Figure 2 details the basic operations illustrated in Figure 1. In the following, we explain this workflow:

1. The Policy Authority generates a pair of public/private keys for Alice, pk_A/sk_A. It sends the public key pk_A to Alice. This could be an offline operation.
2. Bob generates a pair of public/private keys pk_B/sk_B. He sends his public key pk_B to the Policy Authority. The Policy Authority generates a re-encryption key rk_{AB} based on Alice's private key sk_A and Bob's public key pk_B. It sends rk_{AB} to the PRE-enabled broker. This could be an offline operation.
3. Alice tags the message m with topic metadata, encrypts m with her public key pk_A and sends this ciphertext c_A with metadata to the broker.

4. Based on Bob's approved topic subscriptions, the broker re-encrypts the ciphertext c_A with the re-encryption key rk_{AB} such that Bob would be able to decrypt c_B and obtain the original information encrypted by Alice.
5. Bob retrieves the re-encrypted ciphertext c_B from the broker and decrypts it with his secret key sk_B.

The pairs of public/private keys for publishers (generated by the Policy Authority) and for subscribers (self-generated) are normally created before the deployment. However, they could be created on-demand as well.

2.3. Design Considerations

In PICADOR, multiple subscribers may receive information from multiple publishers, and a publisher's information may flow to multiple subscribers, thus supporting one-to-one, many-to-one, one-to-many and many-to-many information flows. The PRE-enabled broker routes content based on the content's topic and the authorized publisher/subscriber pairs. The publishers address messages to a topic (instead of a specific subscriber), and subscribers register with the broker to topics of interest. If authorized, the subscribers can decrypt messages posted by publishers on their topics of interest.

The topic-based addressing information is encoded in metadata that is not encrypted, such as geo-temporal coordinates, keywords, or other relevant information. The encoding is done using domain-specific policy languages [16, 17, 18].

The broker identifies the subscribers using their public keys, which are also associated with their re-encryption keys. To do this, the broker maintains a map from topics to subscribers, which is updated when new subscribers are added to the system. The map is also updated when subscribers add or remove topics of interest. The broker also maintains a map from approved publisher/subscriber pairs to re-encryption keys. When a publisher publishes information on a specific topic, the broker identifies potentially interested subscribers. Then, it looks up the publisher/subscriber pairs to find their re-encryption keys. If entries in the map exist, the broker uses the relevant re-encryption keys to perform the re-encryption operations so that the information can be decrypted by the approved subscribers. The map must also be updated as new re-encryption keys need to be added for new publishers.

In PICADOR, the broker is required to have a unique re-encryption key for each publisher/subscriber pair. This provides the Policy Authority affirmative approval authority to enable communication from individual publishers to individual subscribers. However, this also implies that PICADOR will not scale to the same degree as prior Pub/Sub designs which do not require external approval and re-encryption key generation. It will subsequently take longer to register subscriptions due to the required interaction with the Policy Authority. As such, PICADOR is better suited for applications where subscriber churn is not too high. For example, PICADOR is expected to work well for applications where the broker is loaded with initial re-encryption keys offline before deployment, with additional publishers, subscribers and re-encryption keys being added at a relatively low frequency.

An important design decision is to generate the re-encryption key at the Policy Authority, not at the publisher. We do this in order to improve the decoupling of publishers and subscribers. With this design, publishers and subscribers do not need to know information about each other, which improves the scalability and security of the Pub/Sub system.

Our design pushes trust from the publisher to the Policy Authority, and computational effort and bandwidth requirements from the publisher to the broker. The Policy Authority determines who can share information based on the generation of re-encryption keys. The broker uses information access policies received from the Policy Authority, in combination with the generated re-encryption keys, to determine what subset of information from a publisher should be sent to a subscriber. The publishers and subscribers, who generally have the lowest computational/networking capabilities (e.g., mobile devices), require the lowest computational effort and only need to maintain single keys, thus simplifying deployments.

The Policy Authority is not required to operate and support re-encryption operation online. Furthermore, it may interact with the broker infrequently. Therefore, it could operate over lower-bandwidth communication links, such as satellite, while the communications between the publishers, broker, and subscribers would potentially need to be over higher bandwidth communication links, such as terrestrial radio or wired Ethernet.

PICADOR can be augmented to support message integrity. Each publisher can generate a traditional public/private key pair for digital signatures using RSA or Elliptic Curve Digital Signature Algorithm which are currently included in NSA Suite B [19] as accepted standard secure digital signing techniques. Each message is signed with the publisher's private key and verified by the broker using the publisher's public key. Finally, the PRE scheme guarantees the integrity of the messages received by the subscribers.

3. PRE Cryptosystem

This section presents PICADOR's novel PRE cryptosystem. We start with a discussion of the challenges faced by existing PRE schemes when they are considered in Pub/Sub systems. Next, we overview our solution and explain how it addresses the known challenges of existing PRE schemes. Before covering the theoretical aspects of our solution, we introduce a few concepts of lattice encryption used in the proposed PRE scheme. The section then continues with the formal description of our solution and ends with a discussion of the parameter selection for the proposed PRE scheme.

3.1. Challenges of Existing PRE Schemes

PRE schemes are generally grouped into bidirectional [15, 20, 21] and unidirectional [8, 9, 22, 23, 24]. In bidirectional schemes, a re-encryption key can be used to translate encrypted data not only from the publisher encryption to subscriber encryption but also in reverse, from the subscriber encryption to publisher encryption. Most bidirectional schemes require a pre-sharing of a secret key during re-encryption key generation. Unidirectional schemes guarantee that re-encryption works only one way: it translates publisher ciphertexts (encrypted using the publisher key) into subscriber ciphertexts (that can be decrypted using the private key of the subscriber). Unidirectional schemes usually do not require pre-sharing.

PICADOR requires a unidirectional scheme that does not allow pre-sharing during key generation (model of least trust). Our Pub/Sub system also requires efficient encryption, re-encryption, and decryption operations. Although we focus on operations where information is re-encrypted once (also called one-hop or single-hop re-encryption), our Pub/Sub model generalizes so that ciphertexts can be re-encrypted multiple times (also called multi-hop re-encryption.) The unidirectional schemes presented in [8] require pre-sharing. The scheme proposed in [9] is non-transferable (single-hop) and cannot theoretically support additional hops. The PRE schemes developed in [22, 23, 24] have not been implemented and are based on the Learning With Errors (LWE) constructions that are known to have higher asymptotical complexity (and practical runtime) than lattice constructions based on ideal (cyclic) lattices, i.e., polynomial rings. Therefore, none of the existing unidirectional schemes are suitable for our Pub/Sub model.

3.2. Solution Overview

We propose a novel unidirectional PRE scheme combining the provably secure Stehlé-Steinfeld NTRU scheme [10] and NTRU immunity constraint suggested by Albrecht et al. [25]. The Stehlé-Steinfeld encryption scheme and its security analysis are presented in Appendix A.1 and Appendix A.2, respectively.

Our scheme is based on efficient and secure lattice primitives (ideal lattices). As opposed to other known approaches to PRE, lattice encryption approaches, such as ours, are generally considered post-quantum [11, 12], that is, potentially secure against attacks even from adversaries with practical quantum computing devices in addition to adversaries with classical computing devices [13]. Further, lattice encryption schemes are asymptotically faster than other public-key encryption schemes such as Paillier encryption and RSA.

Another advantage of the proposed PRE scheme is that is based on a partially homomorphic encryption scheme that supports the following homomorphic operations: addition, indexing, and multiplication. Therefore, the Pub/Sub model developed in this paper can be extended to support: (1) addition of encrypted data posted by different publishers, (2) querying specific ciphertext data by indexing, and (3) multiplication of encrypted ciphertexts posted by multiple publishers.

3.3. Lattice Encryption Concepts for Our PRE Scheme

Our approach is a lattice-based cryptographic scheme. A general survey on the design of lattice-based schemes can be found in [26]. Mathematical preliminaries for lattice-based cryptography can be found in [27], but we provide a high-level overview here.

Our lattice-based cryptosystem provides the core public key encryption primitives of Key Generation ($\mathsf{KeyGen} \to (pk, sk)$), Encryption ($\mathsf{Enc}(pk, \mu) \to c$), and Decryption ($\mathsf{Dec}(sk, c) \to \mu$), where pk is a public key, sk is a secret key, μ is a plaintext, and c is a ciphertext.

Concretely, we represent our plaintext and ciphertext as vectors of unsigned integers. We use power-of-2 cyclotomics in our cryptosystem, meaning that plaintext and ciphertext vectors represent the coefficients of polynomials mod $(x^n + 1)$, where n is a power of 2 [27]. Almost all of the operations we need to support on the plaintext and ciphertext are linear transforms and, by restricting ourselves to power-of-2 dimensions, we greatly reduce the implementation complexity required in our cryptosystem. This translates directly into a more efficient implementation. Current implementations designed for non-power-of-2 cyclotomics can support more general capabilities, but we intentionally choose to focus on a simpler scheme for improved runtime for limited-depth applications.

From [27] and the general field of lattice-based encryption, for n a power of 2, we define the ring $R = \mathbb{Z}[x]/\langle x^n + 1\rangle$ (i.e., integer polynomials modulo $x^n + 1$). For any positive integer q, we define the ciphertext space $R_q = R/qR$ (i.e., integer polynomials modulo $x^n + 1$, with mod-q coefficients). The plaintext space is R_p for some integer $p \geq 2$, meaning that plaintext are length-n vectors of integers modulo p.

The practical security of cryptosystems is generally discussed in terms of "work factor", which is the relative computational effort to recover usable information from encrypted data [26]. Work factors are often determined through difficult experimental cryptanalysis of the underlying computational hardness problems used in the proofs of security of cryptosystems. The security of cryptosystems are derived from reduction proofs from the hardness of known hard computational problems for many lattice encryption schemes [26, 27].

Work factors are difficult to translate into meaningful concrete parameter settings. Cryptographers often use a (sometimes artificial) scalar security parameter to encapsulate a representation of the hardness of these problems and the security of the derived cryptosystems. In practice, cryptographers then guarantee the security systems by providing assurances that these security parameters meet a certain minimal value that provides a base level of security.

Security in lattice-based cryptosystems is typically discussed in terms of either the root Hermite factor δ or the bits of security representation [26, 27, 28, 29]. We focus on the root Hermite factor representation of security. Current security estimates indicate that a root Hermite factor $\delta < 1.006$ corresponds to 100 bits of security and an adequate work factor for practical security [28, 29, 30].

3.4. NTRU-RLWE PRE Scheme

Our NTRU-RLWE PRE scheme is parameterized using the following quantities:

- security parameter (root Hermite factor) δ for the corresponding Shortest Vector Problem [28],

- ciphertext modulus q,
- ring dimension n,
- plaintext modulus p,
- B_k-bounded discrete Gaussian (key generation) distribution χ_k over the polynomial ring $R = \mathbb{Z}[n]/\langle x^n + 1\rangle$ with standard deviation σ_k,
- B_e-bounded discrete Gaussian (error) distribution χ_e over the polynomial ring R with standard deviation σ_e,
- relinearization window r to optimize the re-encryption runtime,
- empirically selected assurance measure α to minimize the bit length of q (introduced in our scheme for better performance).

Just like in the case of Stehlé-Steinfeld scheme, we support a plaintext space of $\mathcal{M} = \{0, 1, \ldots, p-1\}^n$, where p is the plaintext modulus. All operations on ciphertexts are performed in the ring $R_q \equiv R/qR$. Each coefficient in the polynomials is represented in the range $\left\{-\left\lfloor \frac{q}{2} \right\rfloor, ..., \left\lfloor \frac{q}{2} \right\rfloor\right\}$.

The scheme includes the following operations:

- KeyGen: Sample polynomials $f', g \leftarrow \chi_k$ and set $f := pf' + 1$ to satisfy $f \equiv 1 \,(\mathrm{mod} p)$. If f has no modular multiplicative inverse in R_q, then re-sample f'. Set the public key pk and private key sk:

$$sk := f \in R,$$

$$pk := h = pg\, f^{-1} \in R_q.$$

- Enc $(pk = h, m \in M)$: Sample polynomials $s, e \leftarrow \chi_e$. Compute the ciphertext:

$$c := hs + pe + m \in R_q.$$

- Dec $(sk = f,\ c)$: Compute the ciphertext error $b := f \cdot c \in R_q$. Output $m' := b \,(\mathrm{mod} p)$.
- ReKeyGen $(pk = h^*, sk = f)$: For every $i = 0, 1, .., \lfloor \log\,(q)\,/r \rfloor$, sample polynomials s_i and e_i and compute

$$\gamma_i = h^* s_i + pe_i +\ f \cdot\ (2^r)^i \in R_q,$$

and set the re-encryption key

$$rk := \gamma = \left(\gamma_0, \gamma_1, ..., \gamma_{\lfloor \log(q)/r \rfloor}\right).$$

- ReEnc $(rk = \gamma, c)$: Compute the ciphertext

$$c' = \sum_{i=0}^{\lfloor \log(q)/r \rfloor} (c_i \cdot \gamma_i),$$

where

$$c_i = \{h \cdot s + pe + m\}_i$$

and

$$c = \sum_{i=0}^{\lfloor \log(q)/r \rfloor} c_i \cdot (2^r)^i.$$

The ReKeyGen and ReEnc operations are based on the relinearization procedure [31] and are defined the same way as in [32]. The scheme is correct as long as there is no wrap-around modulo q until the final stage of decryption. Indeed, for the case of decryption of re-encrypted data we have

$$\begin{aligned} f^* \cdot c' &= f^* \cdot (\sum_i c_i \cdot \gamma_i) \\ &= \sum_i c_i \cdot (f^* \cdot \gamma_i) \\ &= p \sum_i c_i \cdot E_i + \sum_i c_i \, f^* f \cdot (2^r)^i \\ &= p \sum_i c_i \cdot E_i + f^* f \, c, \end{aligned}$$

where

$$E_i = g^* s_i + f^* e_i.$$

It can be seen that

$$f^* \cdot c' = f^* f \, c = m \, (\mathrm{mod} p) \, ,$$

i.e., the decryption is correct, if the ciphertext error $f^* \cdot c'$ is not too large to wrap around q.

Similarly to the derivations in [32], we can devise the correctness constraint for the decryption of the re-encrypted ciphertext using the infinity norms, upper bounds B_k and B_e for the coefficients generated using discrete Gaussian distribution, and average-case analysis of polynomial multiplication based on the Central Limit Theorem. The correctness constraint for single-hop re-encryption requires that

$$f^* \cdot c' = p \sum_i c_i \cdot E_i + f^* f \, c < q/2. \qquad (1)$$

In view of $B_k, B_e >> 1$, the infinity norms can be expressed as

$$\|c_i\|_\infty \leq 2^r - 1,$$

$$\|E_i\|_\infty \leq \sqrt{n} \left(B_k B_e + (pB_k + 1) B_e \right) \approx \sqrt{n} B_k B_e \, (p+1) \, ,$$

$$\|f \, c\|_\infty \leq \sqrt{n} \left(pB_k B_e + p(pB_k + 1) B_e + (pB_k + 1)(p-1) \right) \approx \sqrt{n} p B_k B_e \, (p+1) \, ,$$

$$\|f^* f \, c\|_\infty \leq \sqrt{n} \, (pB_k + 1) \, \|f \, c\|_\infty \approx n p^2 {B_k}^2 B_e \, (p+1) \, .$$

Substituting the expressions for infinity norms into inequality (1), applying approximations based on $B_k, B_e >> 1$, and multiplying the estimated norm bound by a factor of 2 (to compensate for all approximations), we obtain

$$q > 4np^2 B_k B_e \left\{ (2^r - 1) \left(\lfloor \log_2 (q) / r \rfloor + 1 \right) + pB_k \right\}. \quad (2)$$

The bounds B_k and B_e are expressed as $\sigma_k \sqrt{\alpha}$ and $\sigma_e \sqrt{\alpha}$, respectively, where α is an empirical assurance measure introduced in [32]. It corresponds to the probability of $2^{-\alpha+1}$ that a coefficient of discrete Gaussian distribution exceeds the bound B_i, where $i \in \{k, e\}$. With every polynomial multiplication, the probability of exceeding the bound dramatically drops (single-hop proxy re-encryption requires two polynomial multiplications).

Substituting the expressions for bounds into inequality (2) yields

$$q > 4np^2 \alpha \sigma_k \sigma_e \left\{ (2^r - 1) \left(\lfloor \log_2 (q) / r \rfloor + 1 \right) + p\sqrt{\alpha}\sigma_k \right\}. \quad (3)$$

Indistinguishability under Chosen Plaintext Attack (IND-CPA) follows from Theorem 4 in [10]. Note that our scheme uses the constructions from [10], but with the improved NTRU immunity constraint from [25] which arguably preserves the proof of IND-CPA in [10].

The security analysis of the proposed PRE scheme is presented in Appendix A.3.

3.5. Parameter Selection

A general issue with lattice encryption schemes is that they are more complicated to parameterize than other families of encryption schemes. Parameter selection is governed largely by a correctness condition (which is specific to the scheme being analyzed) and security condition(s). The correctness constraint in this case is given by expression (3), and the security conditions are inequalities (A.3) and (A.4) for NTRU and RLWE assumptions, respectively. All three inequalities have to be satisfied by the parameters of the PRE cryptosystem.

Of high importance are the parameters n and q which have the largest direct impact on the runtimes of the scheme. The value of n should be kept as small as possible since runtime is at least linear in n for all operations. The value of q determines the sizes of integers that need to be manipulated computationally, and should generally be kept as small as possible. The inequality (A.4) implies that the value of n grows linearly with the bit length of q.

We begin the process of parameter selection with the security parameter δ, also known as the root Hermite factor. A heuristic argument presented in [28] suggests that a root Hermite factor of $\delta = 1.006$ could provide adequate security (approximately 100-bit security level). Therefore, we select $\delta = 1.006$.

The bound for discrete Gaussian error distribution χ_e is expressed as $B_e = \sigma_e \sqrt{\alpha}$, where σ_e is the standard deviation of the error distribution and α determines the effective probability that a coefficient generated using discrete Gaussian distribution (or a product of discrete Gaussians) exceeds the bound B_e [32]. The value of σ_e is usually chosen in the range from 3 to 6, and we set

Table 1: Effect of plaintext modulus p and relinearization window r on ring dimension n, ciphertext modulus q, and ciphertext expansion factor β

p	r	n	$\log_2 q$	β
2	1	1024	34	34
2	8	1024	38	38
2	16	2048	48	48
16	1	2048	52	13
16	8	2048	52	13
16	16	2048	57	15
256	1	4096	77	10
256	8	4096	77	10
256	16	4096	77	10

the value of σ_e to 4 as in [33]. We set α to 9, which for the case of an integer generated using discrete Gaussian distribution corresponds to the theoretic probability of at most 2^{-8} of choosing a value that exceeds the upper bound B_e. The choice of this value of α was validated empirically for 35,000 iterations of encryptions/decryptions [32]. The same value of α was used for the key generation distribution χ_k.

Subsequent to the selections of δ, σ_e, and α, we can choose n and q experimentally using expressions (A.3), (A.4), and (3) to minimize the runtime/maximize the throughput for various values of the relinearization window r and plaintext modulus p.

Table 1 lists the minimum values of ring dimension n, bit length of ciphertext modulus q, and ciphertext expansion factor β as a function of plaintext modulus p and relinearization window r. The values of p are increased to reduce β (and thus increase the throughput) and the values of r are increased to reduce the computational and space complexity of the re-encryption operation. It can be seen that the ciphertext expansion factor is high for $p = 2$, which implies it should not be used in scenarios where maximum throughput needs to be attained, which is the case for PICADOR. The highest value of r (up to 16) is usually optimal as long as no ring dimension increase or noticeable ciphertext modulus bit length increase is required [32]. The analysis of Table 1 can be summarized as follows:

- lowest encryption/decryption latency is expected for case $p = 2, r = 1$,
- lowest re-encryption latency is expected for case $p = 2, r = 8$,
- highest throughput is expected either for $p = 16, r = 16$ or $p = 256, r = 16$. An experimental evaluation of throughput for $r = 16$ is provided in Section 5.

4. Prototype Architecture and Implementation

We implemented a PICADOR prototype to explore design tradeoffs and to experimentally evaluate our system's performance. The Pub/Sub architecture is implemented in Java, and the PRE encryption is implemented as a C++ lattice encryption library. We implemented all components of the architecture, including the publishers, subscribers, PRE-enabled Pub/Sub broker, and the Policy Authority in a Java-based framework with invokes the C++ encryption library.

In the following, we first discuss the PRE implementation in Section 4.1. Then, we present the Pub/Sub implementation in Section 4.2.

4.1. PRE Implementation

Key generation, encryption, re-encryption, and decryption operations for our PRE scheme are enabled by our PALISADE library for lattice encryption.[2] We selected to build off the PALISADE C++ library because of its well-defined cryptographic APIs and optimized implementation of lattice/arithmetic operations. For convenience, we refer to our implemented NTRU-RLWE PRE scheme using PALISADE as PALISADE-PRE.

Public/private key pairs are generated using PALISADE-PRE's KeyGen API by the Policy Authority (i.e., for publishers) and subscribers. PALISADE-PRE supports serialization and deserialization of the keys and ciphertext as JavaScript Object Notation (JSON) files. PICADOR uses JSON serializations of ciphertext and keys for communication among its components as well as for storage. JSON was chosen because it is lightweight, programmatically accessible, and provides for rapid development and deployment by avoiding the integration and resource requirements of third-party data management systems.

4.2. Pub/Sub Implementation

PICADOR is implemented using Java Server Pages (JSPs), Java Servlets, Java Message Service (JMS), Java Native Interface (JNI) APIs, Message Driven Enterprise Java Beans (MDEJBs), and companion Java classes with all resources running in the Wildfly 9 application server. PICADOR's Java-based components interface with its C++ PALISADE-PRE library using the JNI framework. Java was selected to develop the prototype because of its cross-platform support, reliable APIs for C++ integration via JNI, messaging via JMS, and security support. The JMS-enabled messaging system of Wildfly 9 was chosen to implement PICADOR's component communication because of its rapid deployment, ease of extensibility, performance, and scalability.

Figure 3 presents the main components of our prototype and their communication. The prototype consists of a publisher, a PRE-enabled broker, and a subscriber. Although only one instance of each is illustrated in the figure, there

[2]The PALISADE library is released under a BSD open-source license and is available for download at https://git.njit.edu/palisade/palisade-community-edition.

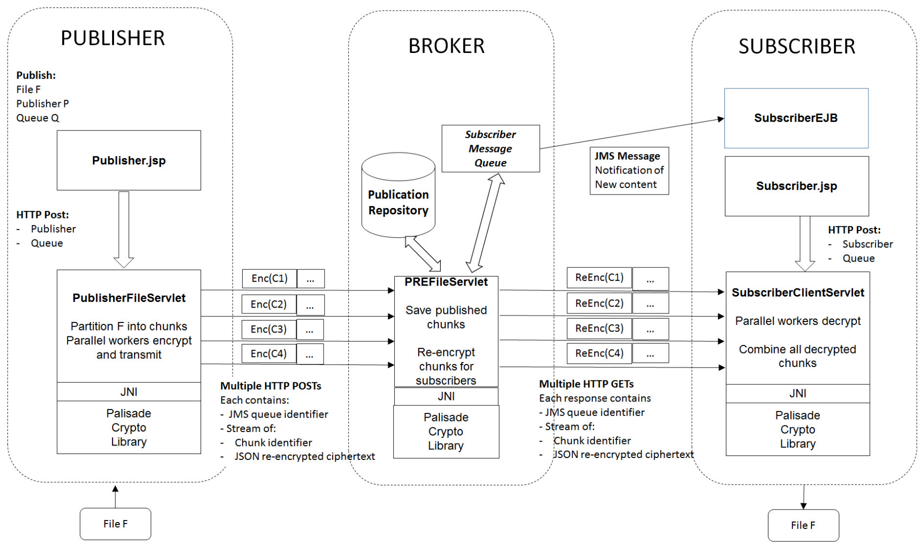

Figure 3: PICADOR Components

can be many instances of the publisher and many instances of the subscriber. Each of these components resides in a Wildfly 9 container. The implementation can be deployed as a single instance of Wildfly with all of the components active and in use, or it can be implemented as multiple instances of Wildfly with each running a subset of the three components. We note that as communication between the components uses HTTP POST and GET, the components can be separated and perhaps in some cases built with smaller implementations should such deployments prove useful.

Although we design, instantiate, and experiment with PICADOR as a single broker Pub/Sub system, it is also possible for our architecture to be generalized to support multiple brokers for improved scalability and fault tolerance. By using distributed systems techniques such as consistent hashing and replication, the topics can be partitioned across multiple brokers while maintaining a given replication factor for per-topic data structures. These data structures are the map between the subscribers and their public keys as well as the map between the publisher/subscriber pairs and the re-encryption keys. Since none of them is updated frequently, the cost of replication and consistency are expected to be low. Since the content is read-only, its replication is straightforward. As demonstrated in [34], messages could be routed in O(1) at the cost of sligthly higher overhead for network topology maintenance. Therefore, the latency is expected to be similar to the one of our current prototype. In this paper, we do not focus on multiple broker designs and we primarily leave this to future work.

The components access the features of the PALISADE-PRE library through the library's JNI wrapper. Each component performs a separate step in the encryption/re-encryption/decryption cycle. Their interaction is through a central content repository and through the shared use of a publisher queue.

The publisher publishes a file by sending encrypted content that is designated for a particular queue to the broker. The queue is specific to the topic of the content. Queues can be added on the fly. The PRE broker stores the published content in the repository, updates the appropriate queue, and notifies any subscribers of updates to the queue. When notified that content has been added to its queue, the subscriber requests a download of the published content. The PRE broker performs re-encryption and sends the re-encrypted content to the subscriber. The subscriber performs decryption, re-assembles the content, and saves the published content file.

4.3. Publisher Workflow

The publisher is accessed through the publisherIndex JSP. The user identifies herself as the publisher, selects (or creates) the queue to which the content is to be published (i.e., the topic), selects the content to be published, and uploads the content to the PublisherFileServlet.

The PublisherFileServlet processes the content by partitioning in into chunks of plaintext. Each of these chunks can be processed separately, allowing the implementation to parallelize the encryption and publication in multiple independent threads. The size of the chunk and the degree of parallelization can be selected by the publisher according to the publisher's available processing capacity; however, as noted, the most efficient chunk size in terms of minimizing the ciphertext expansion factor is a function of n, the PRE ring dimension.

Each thread in the PublisherFileServlet encrypts chunks of plaintext, converts them to a JSON data structure for the ciphertext, and adds a tag for the JSON data structure indicating the sequence of the chunk's position in the overall content. Each thread assembles these tagged JSON structures into an HTTP POST.

The HTTP POST contains a flag informing the broker which topic-based JMS queue is relevant to the published content. The body of the post consists of a sequence of encrypted ciphertext chunks, containing: 1) a portion of the JSON data structure for the ciphertext, and 2) a flag indicating the index of the chunk's JSON portion in the ciphertext's entire JSON data structure. The broker uses those items to store each chunk. In order to make efficient use of network bandwidth, the JSON data structure is compressed before it is sent to the broker using the ZIP compression class in the Java JDK.

When all of the chunks have been processed by the publisher and all of the POST threads have completed, the publisher POSTs a small message indicating that the publication has completed. When the broker receives this "publication complete" POST, it sends a single "publication notice" to the JMS queues it manages for the published content. This notice consists of a content identifier and a flag for the content's JMS queue topic. The notification is implemented using an MDEJB deployed on each subscriber client that is configured to monitor specific queues for notices and stores each notice to the subscriber's publication notice repository (PNR).

4.4. Subscriber Workflow

The subscriber implementation can use the arrival of the publication notice message to launch the process of pulling content that it has subscribed to. However it should be noted that the subscriber may also view the contents of its queue and, at any time, launch the process of pulling content.

The broker maintains a PRE key map that associates each PRE re-encryption key with a specific JMS queue and the subscriber's public key. In this way, the appropriate subscribers can retrieve the content upon receiving the publication notice. Specifically, the subscriber client's JSP interface lists publication notices stored in its PNR that have been received from the JMS queues managed by the broker for the subscriber's subscriptions.

The subscriber client requests content corresponding to a specific notice by sending an HTTP POST to the SubscriberClientServlet. This request contains 1) the subscriber's ID, 2) the topic of the content, and 3) a flag for the relevant JMS queue.

The SubscriberClientServlet submits a request for published content to the PRE Broker. After receiving the subscriber's request, the broker checks to be sure that the subscriber is permitted to receive this information. If the subscriber is authorized, then the broker replies to the request for content by retrieving the content's ciphertext chunks, re-encrypting each chunk, and then sending these PRE ciphertext chunks to the subscriber. Each ciphertext chunk returned contains 1) a PRE ciphertext JSON data structure, and 2) a flag reflecting the index of the chunk in the original ciphertext's entire JSON data structure.

The broker de-compresses the compressed JSON data structure before re-encryption, and then compresses it before sending it to the broker. The subscriber must also de-compress the compressed JSON data structure before decrypting it.

The subscriber decrypts each received PRE ciphertext chunk using its private key to recover a portion of the published content and uses each chunk's index value to reconstruct and store the published content.

The SubscriberClientServlet has the option of parallelizing the fetch of subscribed content from the PRE Broker. This decision can be made independent of the crypto parameters or publication details and is based solely on subscriber resources. This is accomplished by the simple expedient of distributing chunks across threads in a round-robin fashion.

5. Experiments

We experimentally evaluate PICADOR to explore its configuration-performance tradeoffs and scalability, and identify the types of applications it may support. The primary practical challenge of implementing and using lattice-based cryptosystems is one of performance. Typical implementations require tuning of settings such as multi-threading, plaintext encoding, and related parameters, all of which we examine experimentally.

While varying configuration parameters to find optimal settings, we fix the security parameter during experimentation to provide a base level of security which conforms to current best practices and provides a high level of security. We select a security parameter $\delta = 1.006$, which provides a high degree of security based on current understandings [28, 29]. Lowering the security parameter significantly would make the cryptosystem arguably insecure, and increasing the security setting over an acceptable threshold would potentially over-engineer the security of the cryptosystem while negatively impacting runtime performance.

5.1. Methodology

We evaluate performance with respect to the following metrics, which are similar to those used in related PRE work [9, 20]:

1. Throughput: How many plaintext bits can be published or brokered for various settings.
2. Payload Expansion: How many (amortized) bits are required to represent every bit of plaintext (payload).

We use the throughput to assess how much plaintext data can be processed by the implementation per unit of time. Throughput measures the processing time of the specific pub-PRE-sub operations amortized for the number of plaintext bits encrypted in each ciphertext.

Throughput is related to the payload expansion factor. This factor is an artifact of both the ciphertext expansion of the underlying encryption scheme and encoding/decoding performed by the communication infrastructure when transmitting data.

We ran multiple experiments to evaluate the impact on performance of: (i) Plaintext modulus; (ii) Chunk size; (iii) Payload size; (iv) Parallelization in single-host setup; (v) Parallelization in multi-host setup; and (vi) Number of subscribers. All experiments except for the ones dealing with the parallelization in multi-host setup and the effect of the number of subcribers were run on a single host.

The serialized ciphertext data is compressed using GZip before being transmitted to the PRE-enabled broker to reduce bandwidth usage. Once received, the broker decompresses the ciphertext file, performs re-encryption, and then compresses the ciphertext file for the subscriber. We experimentally observed a mean compression ratio in the ciphertext files of under 1.6, meaning that less bandwidth is used during Pub/Sub operations. We observed, however, that the compression and de-compression operations at the broker increases the latency by a mean of 15%. Thus, for settings where the bandwidth is a non-issue and low latency is a high priority, compression may be disabled.

5.2. Testbed

To experimentally evaluate performance, we deployed the PICADOR Pub/Sub broker on a Supermicro 1U rackmount server with an Intel(R) Xeon(R) CPU E5-2660 with 8x2.20 Ghz cores (2 logical processors per each) with 32GB of

Table 2: Effect of Varying Plaintext Modulus on Publisher and Subscriber Throughput

Plaintext Modulus p	Publisher Throughput, KB/s	Subscriber Throughput, KB/s
2	6.4	1.6
16	21.4	4.9
256	33.8	6.4

RAM. The server ran VmWare ESXi 6.0. Virtual machines (VMs) with up to 8 cores and 16 GB of RAM were used to emulate the PRE-enabled broker functionality. The VMs were configured with Windows 7 64-bit.

We used commodity Dell Latitude laptops to emulate publishers and subscribers in multi-host experiments. The laptops had Intel(R) Core(TM) i7-3520M CPU with 2x2.90GHz cores (2 logical processors per core) and 8GB of RAM. The laptops were configured with Windows 7 64-bit.

The Pub/Sub clients connected to the server via a wired Ethernet switch with 100Mbps ports. We use the term "Publisher Throughput" to denote the rate at which plaintext data is encrypted and published to the Pub/Sub broker, and the term "Subscriber Throughput" to denote the rate at which this plaintext data is received at a subscriber, inclusive of it being re-encrypted, sent to the subscriber and then decrypted by the subscriber.

5.3. Plaintext Modulus

Plaintext could hypothetically impact performance because larger plaintext moduli are likely to enable more efficient packing of data into ciphertexts that need to be transmitted. We experimentally evaluated this hypothesis over multiple plaintext selections: $p \in \{2, 16, 256\}$. The results of the impact on publisher and subscriber throughput can be seen in Table 2.

We observe that the throughput generally increases with larger plaintext moduli. This supports our hypothesis that larger moduli could lead to more effective data packing. However, there is a limit to how large of a plaintext modulus we would wish to use. As the plaintext moduli increase, the ciphertext moduli have to correspondingly increase to guarantee correct decryption. When ciphertext moduli increase, the ring dimension also increases as a result of the RLWE security constraint. Eventually a point is reached where further plaintext modulus increase starts causing performance decline.

The prototype plaintext encoding implementation supports the plaintext modulus of up to $p = 256$ for byte encoding of plaintext. Further analysis suggests that increasing p to higher values may provide additional performance improvement, but the net effect would also depend on the value of relinearization window. This type of multi-parameteric optimization based on plaintext modulus and relinearization window is beyond the scope of this study and should be carried out when specific application requirements are known.

5.4. Effect of Chunk Size on Performance

We experimentally evaluated the impact of chunk size on performance. We ran experiments with chunk sizes varied from 4KB up to 1024KB and all other

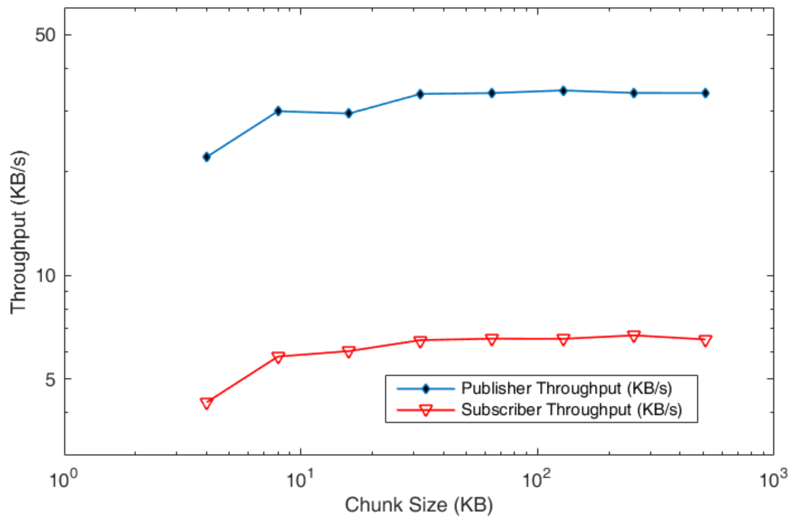

Figure 4: Effect of Chunk Size on Throughput

variables held constant. The experimental results are shown in Figure 4.

The obtained throughput is usable for applications handling text or small size data where low latency is needed. The values suggest that photos or other types of larger data can also be supported by PICADOR, but within applications that do not have stringent latency constraints. Furthermore, Section 5.6 shows that throughput can be increased significantly through parallelization. Therefore, we conclude that many useful applications in domains such as healthcare and military can guarantee end-to-end payload confidentiality with PICADOR, while achieving good performance.

We observe that publisher and subscriber throughput reaches a plateau for chunk sizes higher than 32KB. Lower throughputs for chunk size less than 32KB are likely caused by internal pre-processing steps needed by the C++ wrapper for crypto operations. The wrapper implementation can be optimized to reduce this effect, but subscriber operations are much slower than the publishing step for the single-threaded setup. This is due to the fact that re-encryption, which is the limiting latency factor, is part of the subscriber operations. Our conclusion, based on these experimental observations, is that the chunk size should be large enough to adequately amortize the start-up costs, but not larger than that. Larger chunk sizes are likely to cause latency issues, which may be problematic in safety-critical or mobile applications.

5.5. Effect of Payload Size on Performance

We made experimental observations for payloads varying from 100 KB to 50 MB, with all the other parameters held constant.

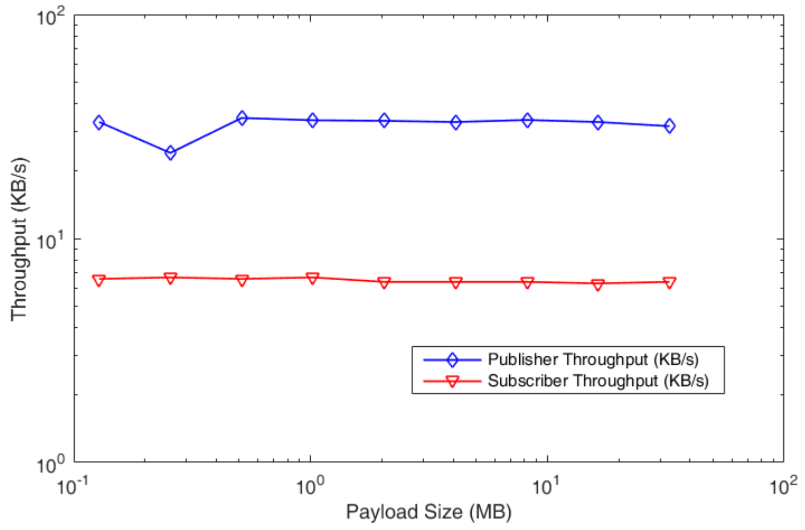

Figure 5: Effect of Payload Size on Throughput

The results from this experiment can be seen in Figure 5, and they confirm that PICADOR scales well with the payload size. The figure shows that both publisher and subscriber throughputs are generally independent of payload size except for very small payload sizes, where the start-up cost is not amortized. This lack of dependence is likely due to the stream-oriented processing of data (as chunks), which allows for good scalability in PICADOR.

5.6. Effect of Parallelization on Performance in Single-Host Setup

These experiments evaluate the impact of multi-threading on the throughput. As we described in Section 4, publishers and subscribers use multiple threads for parallel uploads and downloads to/from the broker. With our baseline parameter set and a payload size fixed at 10MB, we varied the number threads available to the PRE-enabled broker from 1 to 16.

The results, shown in Figure 6, demonstrate significant increase in the throughput as the number of threads increases. The publish throughput increases by 5.9 times, while the subscribe throughput increases by 5.8 times. However, the throughput levels off when we increase the number of threads beyond 8 because our server has 8 cores. Although not shown in the graph, the encryption time for 8 threads is reduced by a factor of 6.3.

5.7. Effect of Parallelization on Performance in Multi-Host Setup

We assessed the impact of multi-core operations on throughput performance when commodity laptops are used to emulate the Pub/Sub clients, and the server virtual machine is operated as a PRE broker. Although PICADOR

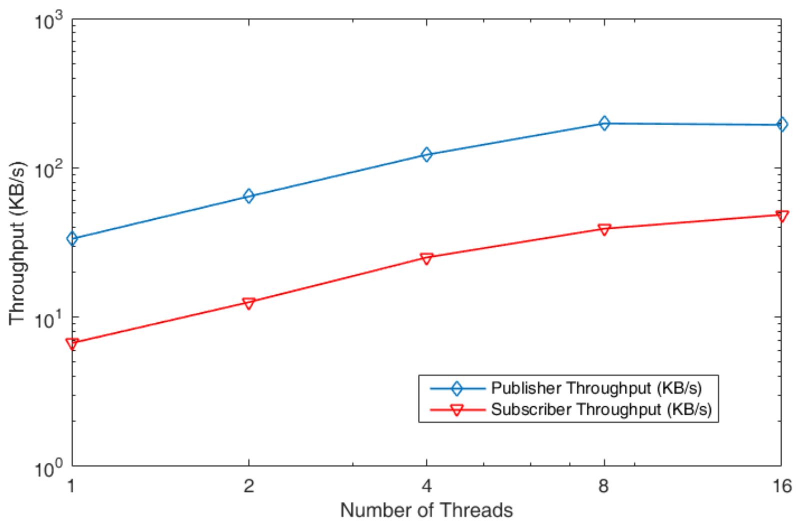

Figure 6: Effect of Number of Running Threads on Throughput

Table 3: Effect of Multi-Core Execution on Throughput

Number of Threads per Client	Number of Server Cores	Publisher Throughput (KB/s)	Subscriber Throughput (KB/s)
4	1	94.9	3.5
4	2	95.7	6.6
4	4	95.4	16.0
4	8	95.1	23.7
2	8	72.8	16.4
1	8	41.5	9.4

is designed for interleaved publish/subscribe operations, we performed non-interleaved publish and subscribe operations to directly measure the effect of each operation separately. Two laptop clients were either concurrently publishing content or downloading the content through subscription to a single PRE broker.

The number of threads was varied to examine both the effects of parallelization and scalability. The actual number of cores on the PRE-enabled broker was changed using the settings of the virtual machine to examine scalability. The experimental results can be seen in Table 3.

We observe that publisher throughput depends only on the performance of publisher clients and not the broker. This is justified because the computational bottleneck is on the publisher client due to encryption operations. When the number of publisher client threads on the 4-core host laptop is increased from 1 to 4, the throughput increases by a factor of 2.3.

Conversely, the subscriber throughput depends both on the computing power of the broker and the subscriber clients. It can be seen that subscriber through-

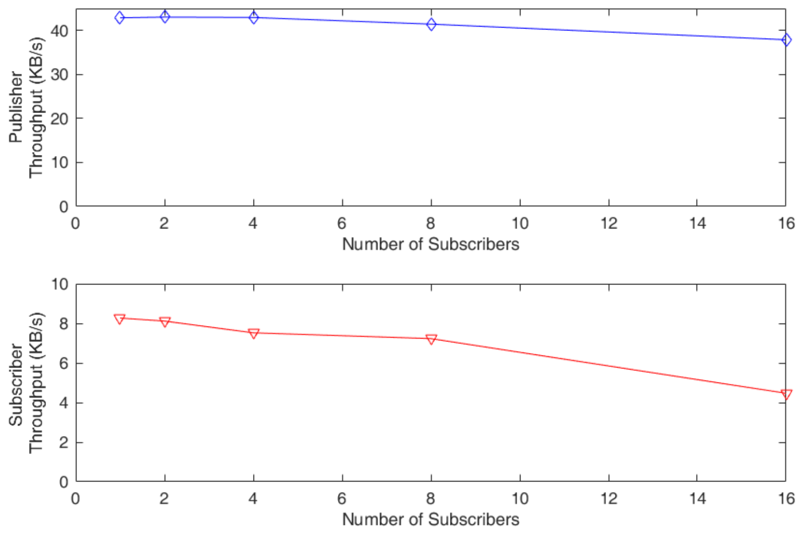

Figure 7: Effect of Number of Subscribers on Throughput

put increases by a factor of 6.8 when the number of server cores is increased from 1 to 8. There is a clear effect of thread synchronization between client and broker. When the number of client threads is one and PRE server cores is 8, only two cores (one for each client) are used by the broker.

5.8. Effect of Number of Subscribers on Performance in Multi-Host Setup

We conclude the experiments by examining the scalability of the PRE server as a function of the number of subscribers. As a base configuration for the PRE server, we used an 8-core virtual machine. All Pub/Sub operations were run in the single-threaded mode. We interleaved the publish and subscribe operations to examine the scalability limits. The subscriber workflows were launched simultaneously to examine the effect of concurrent subscriber sessions. We used the chunk size and payload size of 128 KB and 10 MB, respectively.

Figure 7 shows that the PRE server scales relatively well up to 8 subscribers, which is the same as the number of cores in the PRE server, with unit subscriber throughput reduced only by 13% when the number of concurrent subscribers increases from 1 to 8. When the number of subscribers rises to 16, the throughput degrades by more than 38% as compared to the case of 8 subscribers. This suggests that the number of cores in the PRE server is the main scalability bottleneck in our experimental setup. One can also observe that the effect of high PRE server CPU utilization (due to a large number of subscribers) on the publisher throughput is relatively small as all computationally expensive steps are performed by the publisher. For instance, the publisher throughput decreases only by 12% when the number of subscribers, which corresponds to the number of running threads on the PRE server, is increased from 1 to 16.

We validated the above conclusions by performing additional experiments where the number of cores in the PRE server or the number of publishers was varied.

5.9. Analysis and Tradeoff Recommendations

Based on our experimental results, we have identified the following potential performance bottlenecks that should be considered when applying PICADOR or designing a similar system:

- Payload expansion: one needs to set parameters, including plaintext modulus, ciphertext modulus, and chunk size to minimize payload expansion. This greatly reduces bandwidth usage and reduces computational overhead associated with encryption operations.
- Brokering latency: related to payload expansion, brokering latency is driven by both payload expansion and computational resources at the broker. In practice, this can be addressed by multi-threading on multi-core servers because our PRE scheme supports re-encrypting data chunks in parallel.
- Scalability: Our current architecture is centralized and it has scalability limits. However, as discussed in Section 4, a scalable multi-broker version of PICADOR could be designed using standard distributed systems techniques such as consistent hashing and replication. This version is expected to scale because the overhead associated with maintaining consistency should be low (i.e., new subscribers or new topics are not added very frequently).
- Subscription churn: PICADOR was not designed for highly volatile environment with high subscription churn. To support such a scenario in a scalable way, a part of PICADOR's architecture, including the Policy Authority, would need to be re-designed.

Based on our experimental analysis and identification of bottlenecks, we make the following recommendations regarding the configuration of the PICADOR prototype:

- Use as large a plaintext modulus as possible as long as the ciphertext modulus bitwidth is less than the bitwidth of all processors on the clients or the broker.
- Make no adjustments for varied payload size.
- Use a large enough chunk size to amortize for crypto operation start-up costs, if any.
- Use as much multi-threading as possible, with the limiting factor being the number of cores on the machines.

6. Related Work

Although Pub/Sub information distribution has been actively researched and used [1], there has been limited investigation into the security of Pub/Sub operations. General security challenges in such systems have been identified in [6, 35], particularly confidentiality and key management which are the focus of our contributions in this article. Prior overviews of the challenges faced by cryptographic solutions for Pub/Sub operations are presented in [36, 37]. The system discussed in [38] presents an approach to update publisher encryption keys in content-based Pub/Sub systems for encrypted subscriptions. PICADOR does not address this prior more general capability. PICADOR focuses on topic-based Pub/Sub systems where subscription information is not encrypted, and on key exchange rather than on key updates as in [38].

Confidentiality and other security concerns in topic-based Pub/Sub systems such as PICADOR have potentially different security concerns and trade-offs as compared to content-based Pub/Sub systems. There has been extensive prior work on confidentiality in content-based Pub/Sub frameworks where the broker needs some degree of visbility into content to support brokering [39, 40, 41]. A solution based on symmetric key encryption, shown in [39], requires either decryption at the Pub/Sub broker or establishing a priori shared symmetric keys between publishers and subscribers. A general approach to confidential Pub/Sub systems with a highly scalable key-management solution, presented in [40], can be overlaid on general content-based Pub/Sub information substrates.

PICADOR focuses on preserving the privacy of the published content; it does not address the privacy of topic metadata. However, solutions presented in [14, 41] provide insight into an approach which protects the confidentiality of subscriptions, potentially by encrypting subscription information. The PRE techniques in PICADOR could be combined with techniques in [14] to protect the privacy of subscriptions in addition to the confidentiality of the information payloads.

A prior secure topic-based Pub/Sub design [6] relies on symmetric encryption to enforce end-to-end content confidentiality and focuses on identity management to ensure that only approved subscribers recieve information. Also, [42] provides a related topic-based publish-subscribe framework that focuses on identity management with end-to-end encryption. These works are highly complimentary to PICADOR in that they addresses identity management issues that are not our focus.

There is a growing relevant body of work which applies Attribute-Based Encryption (ABE) [43] to support relevant ABE-based Access Control [44] and ABE-based Pub/Sub [45, 46]. ABE-based approaches are appealing because the encryption of data depends on attributes of intended data recipients, inclusive of which topics the recipients are interested in, without being tied to specific recipients. This line of research shows potential promise to provide additional confidentiality of topics, although it might not yet be completely general and practical. A general challenge with ABE-based approaches is the complexity of their encryption operations which may be difficult to support efficiently on limited publisher hardware such as the one encountered on mobile devices. The ABE performance limitations are addressed in [45, 46], but the ABE approaches generally require substantially more computational effort and publisher bandwidth than our PRE-based approach. In addition, ABE does not provide the general access delegation capabilities we exploit with PRE, which features encrypted access delegation.

There has been limited application of Proxy Re-Encryption (PRE) in the domain of Publish-Subscribe systems. Most relevant is [47] which was not implemented and uses a Paillier-based PRE scheme [48] rather than a lattice-based approach employed in our work. An important feature of our lattice-based PRE is that it is *unidirectional* and provides tighter control on which ciphertexts can be re-encrypted and to whom. Furthermore, our lattice-based scheme is post-quantum and admits more general homomorphic encryption properties.

The work related to PRE is discussed in detail at the beginning of Section 3.

7. Conclusions and Future Work

We presented PICADOR, a secure topic-based Pub/Sub information architecture with end-to-end encryption that delegates information access with a lattice-based variant of Proxy Re-Encryption (PRE). PICADOR enables flexible Pub/Sub asynchronous communication because it prevents the need for information producers and consumers to a priori share information. Furthermore, it ensures the Pub/Sub broker does not have access to the unprotected information. Experimental results demonstrate that PICADOR is practically viable. It provides adequate performance for many applications that do not have stringent real-time requirements, while transferring large amounts of data.

An important benefit of PICADOR's end-to-end lattice-based encryption scheme is that it enables users to securely use low-cost cloud computing environments to share data while also significantly reducing vulnerability to insider attacks. For example, our architecture limits data access only to the system administrators who have decryption keys even when encrypted computing is hosted on proprietary servers. Additionally, our architecture prevents access to decrypted data until it reaches its intended recipient. The benefits of our architecture will thus reduce the operational costs of highly regulated industries such as healthcare and military where regulatory compliance has previously restricted the ability to outsource data security computations to low-cost cloud computing environments.

Although we focused on privacy and confidentiality concerns, our architecture can be augmented to provide integrity and availability protections thereby satisfying the CIA triad. For example, standard digital signature methods, such as RSA signing or Elliptic Curve Digital Signature Algorithm, could satisfy the integrity requirement. Service replication, for example, implemented using Active/Active Clustering, could satisfy the availability requirement against hardware issues and denial-of-service attacks. The prototype's Policy Authority-centric key distribution scheme can also be augmented to allow the authentication of transmitted public and PRE keys.

The PICADOR implementation is based on the NTRU-RLWE PRE scheme that supports up to two hops of re-encryption (two encrypted-data delegations or two brokers connected in series). If a larger number of re-encryptions is required, other lattice encryption schemes, such as BGV [30] and FV [49], can be used. These schemes can be implemented using the PALISADE C++ library employed for the PICADOR prototype.

Acknowledgement

The authors wish to acknowledge conversations with Prof. Shafi Goldwasser and Prof. Vinod Vaikuntanathan from MIT which supported this work.

References

[1] P. T. Eugster, P. A. Felber, R. Guerraoui, A.-M. Kermarrec, The many faces of publish/subscribe, ACM Comput. Surv. 35 (2) (2003) 114–131. doi:10.1145/857076.857078.
URL http://doi.acm.org/10.1145/857076.857078

[2] A. King, S. Procter, D. Andresen, J. Hatcliff, S. Warren, W. Spees, R. Jetley, P. Jones, S. Weininger, A publish-subscribe architecture and component-based programming model for medical device interoperability, SIGBED Rev. 6 (2) (2009) 7:1–7:10. doi:10.1145/1859823.1859830.
URL http://doi.acm.org/10.1145/1859823.1859830

[3] A. King, S. Procter, D. Andresen, J. Hatcliff, S. Warren, W. Spees, R. Jetley, P. Jones, S. Weininger, An open test bed for medical device integration and coordination, in: Software Engineering - Companion Volume, 2009. ICSE-Companion 2009. 31st International Conference on, 2009, pp. 141–151. doi:10.1109/ICSE-COMPANION.2009.5070972.

[4] V. T. Combs, M. H. Linderman, Jini-based publish and subscribe capability, Proc. SPIE 4863 (2002) 59–69. doi:10.1117/12.472945.
URL http://dx.doi.org/10.1117/12.472945

[5] J. P. Loyall, M. Carvalho, A. Martignoni III, D. Schmidt, A. Sinclair, M. Gillen, J. Edmondson, L. Bunch, D. Corman, QoS enabled dissemination of managed information objects in a publish-subscribe-query information broker, Proc. SPIE 7350 (2009) 73500M–73500M–12. doi:10.1117/12.818744.
URL http://dx.doi.org/10.1117/12.818744

[6] S. Pallickara, M. Pierce, H. Gadgil, G. Fox, Y. Yan, Y. Huang, A framework for secure end-to-end delivery of messages in publish/subscribe systems, in: Proceedings of the 7th IEEE/ACM International Conference on Grid Computing, GRID '06, IEEE Computer Society, Washington, DC, USA, 2006, pp. 215–222. doi:10.1109/ICGRID.2006.311018.
URL http://dx.doi.org/10.1109/ICGRID.2006.311018

[7] E. Orakwue, Private clouds: Secure managed services, Information Security Journal: A Global Perspective 19 (6) (2010) 295–298. arXiv:http://dx.doi.org/10.1080/19393550903482924, doi:10.1080/19393550903482924.
URL http://dx.doi.org/10.1080/19393550903482924

[8] A.-A. Ivan, Y. Dodis, Proxy cryptography revisited., in: in Proceedings of the Network and Distributed System Security Symposium (NDSS), 2003.

[9] G. Ateniese, K. Fu, M. Green, S. Hohenberger, Improved proxy re-encryption schemes with applications to secure distributed storage, ACM Transactions on Information and System Security (TISSEC) 9 (1) (2006) 1–30.

[10] D. Stehlé, R. Steinfeld, Making NTRU as secure as worst-case problems over ideal lattices, in: K. G. Paterson (Ed.), Advances in Cryptology – EUROCRYPT 2011: 30th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tallinn, Estonia, May 15-19, 2011. Proceedings, Springer Berlin Heidelberg, Berlin, Heidelberg, 2011, pp. 27–47.

[11] D. Micciancio, O. Regev, Lattice-based cryptography, in: Post Quantum Cryptography, Springer, 2009, pp. 147–191.

[12] O. Regev, Quantum computation and lattice problems, SIAM J. Comput. 33 (3) (2004) 738–760, preliminary version in FOCS 2002.

[13] D. Micciancio, Lattice-based cryptography, in: Encyclopedia of Cryptography and Security, Springer, 2011, pp. 713–715.

[14] R. Barazzutti, P. Felber, H. Mercier, E. Onica, E. Rivière, Thrifty privacy: Efficient support for privacy-preserving publish/subscribe, in: Proceedings of the 6th ACM International Conference on Distributed Event-Based Systems, DEBS '12, ACM, New York, NY, USA, 2012, pp. 225–236. doi:10.1145/2335484.2335509.
URL http://doi.acm.org/10.1145/2335484.2335509

[15] M. Blaze, G. Bleumer, M. Strauss, Divertible protocols and atomic proxy cryptography, in: Advances in CryptologyEUROCRYPT'98, Springer, 1998, pp. 127–144.

[16] Z. Miklos, Towards an access control mechanism for wide-area publish/subscribe systems, in: Distributed Computing Systems Workshops, 2002. Proceedings. 22nd International Conference on, 2002, pp. 516–521. doi:10.1109/ICDCSW.2002.1030820.

[17] A. Belokosztolszki, D. M. Eyers, P. R. Pietzuch, J. Bacon, K. Moody, Role-based access control for publish/subscribe middleware architectures, in: Proceedings of the 2Nd International Workshop on Distributed Event-based Systems, DEBS '03, ACM, New York, NY, USA, 2003, pp. 1–8.

doi:10.1145/966618.966622.
URL http://doi.acm.org/10.1145/966618.966622

[18] J. Hoffert, D. Schmidt, A. Gokhale, A QoS policy configuration modeling language for publish/subscribe middleware platforms, in: Proceedings of the 2007 Inaugural International Conference on Distributed Event-based Systems, DEBS '07, ACM, New York, NY, USA, 2007, pp. 140–145. doi:10.1145/1266894.1266922.
URL http://doi.acm.org/10.1145/1266894.1266922

[19] Information Technology Laboratory, FIPS Pub 186-4 Digital Signature Standard (DSS), National Institute of Standards and Technology Federal Information Processing Standards Publication.

[20] D. Nuñez, I. Agudo, J. Lopez, NTRUReEncrypt: An efficient proxy re-encryption scheme based on NTRU, in: Proceedings of the 10th ACM Symposium on Information, Computer and Communications Security, ACM, 2015, pp. 179–189.

[21] R. Canetti, S. Hohenberger, Chosen-ciphertext secure proxy re-encryption, in: Proceedings of the 14th ACM Conference on Computer and Communications Security, CCS '07, ACM, New York, NY, USA, 2007, pp. 185–194. doi:10.1145/1315245.1315269.
URL http://doi.acm.org/10.1145/1315245.1315269

[22] Y. Aono, X. Boyen, L. Wang, et al., Key-private proxy re-encryption under LWE, in: Progress in Cryptology–INDOCRYPT 2013, Springer, 2013, pp. 1–18.

[23] E. Kirshanova, Proxy re-encryption from lattices, in: Public-Key Cryptography–PKC 2014, Springer, 2014, pp. 77–94.

[24] X. Fan, F.-H. Liu, Various proxy re-encryption schemes from lattices, Cryptology ePrint Archive, Report 2016/278, http://eprint.iacr.org/ (2016).

[25] L. D. Martin Albrecht, Shi Bai, A subfield lattice attack on overstretched NTRU assumptions: Cryptanalysis of some FHE and graded encoding schemes, Cryptology ePrint Archive, Report 2016/127, http://eprint.iacr.org/ (2016).

[26] C. Peikert, A decade of lattice cryptography, Foundations and Trends® in Theoretical Computer Science 10 (4) (2016) 283–424.

[27] V. Lyubashevsky, C. Peikert, O. Regev, A toolkit for ring-LWE cryptography., in: EUROCRYPT, Vol. 7881, Springer, 2013, pp. 35–54.

[28] Y. Chen, P. Q. Nguyen, BKZ 2.0: Better lattice security estimates, in: ASIACRYPT, Vol. 7073 of Lecture Notes in Computer Science, Springer, 2011, pp. 1–20.

[29] T. Lepoint, M. Naehrig, A Comparison of the Homomorphic Encryption Schemes FV and YASHE, Springer International Publishing, Cham, 2014, pp. 318–335.

[30] C. Gentry, S. Halevi, N. P. Smart, Homomorphic evaluation of the AES circuit, in: Advances in Cryptology–CRYPTO 2012, Springer, 2012, pp. 850–867.

[31] A. López-Alt, E. Tromer, V. Vaikuntanathan, On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption, in: STOC, 2012, pp. 1219–1234.

[32] Y. Polyakov, K. Rohloff, G. Sahu, V. Vaikuntanathan, Fast proxy re-encryption for publish/subscribe systems, Submitted.

[33] C. Gentry, S. Halevi, N. Smart, Homomorphic evaluation of the AES circuit, in: R. Safavi-Naini, R. Canetti (Eds.), Advances in Cryptology CRYPTO 2012, Vol. 7417 of Lecture Notes in Computer Science, Springer Berlin / Heidelberg, 2012, pp. 850–867.

[34] G. DeCandia, D. Hastorun, M. Jampani, G. Kakulapati, A. Lakshman, A. Pilchin, S. Sivasubramanian, P. Vosshall, W. Vogels, Dynamo: Amazon's highly available key-value store, in: Proceedings of Twenty-first ACM SIGOPS Symposium on Operating Systems Principles, SOSP '07, 2007, pp. 205–220.

[35] A. Rowstron, A.-M. Kermarrec, M. Castro, P. Druschel, Scribe: The Design of a Large-Scale Event Notification Infrastructure, Springer Berlin Heidelberg, Berlin, Heidelberg, 2001, pp. 30–43. `doi:10.1007/3-540-45546-9_3`.
URL `http://dx.doi.org/10.1007/3-540-45546-9_3`

[36] P. Nikander, G. F. Marias, Towards understanding pure publish/subscribe cryptographic protocols, in: B. Christianson, J. A. Malcolm, V. Matyas, M. Roe (Eds.), Security Protocols XVI: 16th International Workshop, Cambridge, UK, April 16-18, 2008. Revised Selected Papers, Springer Berlin Heidelberg, Berlin, Heidelberg, 2011, pp. 144–155. `doi:10.1007/978-3-642-22137-8_21`.
URL `http://dx.doi.org/10.1007/978-3-642-22137-8_21`

[37] T. H. Yuen, W. Susilo, Y. Mu, Towards a cryptographic treatment of publish/subscribe systems, J. Comput. Secur. 22 (1) (2014) 33–67.
URL `http://dl.acm.org/citation.cfm?id=2590636.2590638`

[38] E. Onica, P. Felber, H. Mercier, E. Rivière, Efficient key updates through subscription re-encryption for privacy-preserving publish/subscribe, in: Proceedings of the 16th Annual Middleware Conference, Middleware '15, ACM, New York, NY, USA, 2015, pp. 25–36. `doi:10.1145/2814576.2814805`.
URL `http://doi.acm.org/10.1145/2814576.2814805`

[39] H. Khurana, Scalable security and accounting services for content-based publish/subscribe systems, in: Proceedings of the 2005 ACM Symposium on Applied Computing, SAC '05, ACM, New York, NY, USA, 2005, pp. 801–807. doi:10.1145/1066677.1066862.
URL http://doi.acm.org/10.1145/1066677.1066862

[40] M. Srivatsa, L. Liu, A. Iyengar, Eventguard: A system architecture for securing publish-subscribe networks, ACM Trans. Comput. Syst. 29 (4) (2011) 10:1–10:40. doi:10.1145/2063509.2063510.
URL http://doi.acm.org/10.1145/2063509.2063510

[41] M. Nabeel, N. Shang, E. Bertino, Efficient privacy preserving content based publish subscribe systems, in: Proceedings of the 17th ACM Symposium on Access Control Models and Technologies, SACMAT '12, ACM, New York, NY, USA, 2012, pp. 133–144. doi:10.1145/2295136.2295164.
URL http://doi.acm.org/10.1145/2295136.2295164

[42] M. A. Rajan, A. Varghese, N. Narendra, M. Singh, V. L. Shivraj, G. Chandra, B. P., Security and privacy for real time video streaming using hierarchical inner product encryption based publish-subscribe architecture, in: 2016 30th International Conference on Advanced Information Networking and Applications Workshops (WAINA), 2016, pp. 373–380. doi:10.1109/WAINA.2016.101.

[43] J. Bethencourt, A. Sahai, B. Waters, Ciphertext-policy attribute-based encryption, in: 2007 IEEE Symposium on Security and Privacy (SP '07), 2007, pp. 321–334. doi:10.1109/SP.2007.11.

[44] V. Goyal, O. Pandey, A. Sahai, B. Waters, Attribute-based encryption for fine-grained access control of encrypted data, in: Proceedings of the 13th ACM Conference on Computer and Communications Security, CCS '06, ACM, New York, NY, USA, 2006, pp. 89–98. doi:10.1145/1180405.1180418.
URL http://doi.acm.org/10.1145/1180405.1180418

[45] M. Ion, G. Russello, B. Crispo, Providing confidentiality in content-based publish/subscribe systems, in: Security and Cryptography (SECRYPT), Proceedings of the 2010 International Conference on, IEEE, 2010, pp. 1–6.

[46] P. Pal, G. Lauer, J. Khoury, N. Hoff, J. Loyall, P3S: A privacy preserving publish-subscribe middleware, in: P. Narasimhan, P. Triantafillou (Eds.), Middleware 2012: ACM/IFIP/USENIX 13th International Middleware Conference, Montreal, QC, Canada, December 3-7, 2012. Proceedings, Springer Berlin Heidelberg, Berlin, Heidelberg, 2012, pp. 476–495. doi:10.1007/978-3-642-35170-9_24.
URL http://dx.doi.org/10.1007/978-3-642-35170-9_24

[47] K. R. Rohloff, M. J. Gillen, J. P. Loyall, Information management using proxy re-encryption, US Patent Application US2015/0271153 A1 (Sep. 24 2015).

[48] G. Ateniese, K. Fu, M. Green, S. Hohenberger, Improved proxy re-encryption schemes with applications to secure distributed storage, ACM Trans. Inf. Syst. Secur. 9 (1) (2006) 1–30. `doi:10.1145/1127345.1127346`. URL `http://doi.acm.org/10.1145/1127345.1127346`

[49] J. Fan, F. Vercauteren, Somewhat practical fully homomorphic encryption, Cryptology ePrint Archive, Report 2012/144, `http://eprint.iacr.org/` (2012).

[50] C. L. Jung Hee Cheon, Jinhyuck Jeong, An algorithm for NTRU problems and cryptanalysis of the GGH multilinear map without a low level encoding of zero, Cryptology ePrint Archive, Report 2016/139, `http://eprint.iacr.org/` (2016).

Appendix A.

Appendix A.1. Stehlé-Steinfeld Encryption Scheme

Our PRE scheme is based on the Stehlé-Steinfeld encryption scheme [10], which is summarized below. The Stehlé-Steinfeld scheme is parameterized using the following quantities:

- security parameter (root Hermite factor) δ for the corresponding Shortest Vector Problem [28],
- ciphertext modulus q,
- ring dimension n,
- plaintext modulus p,
- B_k-bounded discrete Gaussian (key generation) distribution χ_k over the polynomial ring $R = \mathbb{Z}[n]/\langle x^n + 1\rangle$ with standard deviation σ_k,
- B_e-bounded discrete Gaussian (error) distribution χ_e over the polynomial ring R with standard deviation σ_e.

The scheme supports a plaintext space of $\mathcal{M} = \{0, 1, \dots, p-1\}^n$, where p is the plaintext modulus. All operations on ciphertexts are performed in the ring $R_q \equiv R/qR$. Each coefficient in the polynomials is represented in the range $\left\{-\left\lfloor \frac{q}{2} \right\rfloor, ..., \left\lfloor \frac{q}{2} \right\rfloor\right\}$.

The scheme includes the following operations:

- KeyGen: Sample polynomials $f', g \leftarrow \chi_k$ and set $f := pf' + 1$ to satisfy $f \equiv 1 \,(\mathrm{mod} p)$. If f has no modular multiplicative inverse in R_q, then re-sample f'. Set the public key pk and private key sk:

$$sk := f \in R,$$

$$pk := h = pg\, f^{-1} \in R_q.$$

- $\mathsf{Enc}\,(pk = h, m \in M)$: Sample polynomials $s, e \leftarrow \chi_e$. Compute the ciphertext:
$$c := hs + pe + m \in R_q.$$
- $\mathsf{Dec}\,(sk = f,\ c)$: Compute the ciphertext error $b := f \cdot c \in R_q$. Output $m' := b\,(\mathrm{mod} p)$.

The scheme is correct as long as there is no wrap-around modulo q. Indeed,

$$b = f \cdot c = f\,(h\,s + pe + m)\ = pgs + pfe + fm$$

and if the value of b does not wrap around modulo q, then

$$b = pgs + pfe + fm\ (\mathrm{mod} p) = fm\ (\mathrm{mod} p) = m\ (\mathrm{mod} p).$$

Similarly to the derivations in [32, 31], one can devise the correctness constraint for the decryption of ciphertext using the infinity norms $\|...\|_\infty$, upper bounds B_k and B_e for the coefficients generated using discrete Gaussian distribution, and average-case analysis of polynomial multiplication based on the Central Limit Theorem. The correctness constraint for the encryption scheme requires that
$$f \cdot c = pgs + pfe + fm < q/2. \tag{A.1}$$

In view of $B_k, B_e >> 1$, the infinity norm can be expressed as

$$\|f \cdot c\|_\infty \leq \sqrt{n}\,\{pB_kB_e + p\,(pB_k + 1)\,B_e + (pB_k + 1)\,(p - 1)\} \approx$$
$$\sqrt{n}p\,(p + 1)\,B_kB_e.$$

Substituting the expression for infinity norm into inequality (A.1) and multiplying the estimated norm bound by a factor of 2 (to compensate for all approximations), we obtain

$$q > 4\sqrt{n}p\,(p + 1)\,B_kB_e. \tag{A.2}$$

Appendix A.2. Security Analysis of Stehlé-Steinfeld Encryption Scheme

The security of the Stehlé-Steinfeld scheme is based on the NTRU and "Ring Learning With Errors" (RLWE) assumptions [10, 31].

The NTRU problem is to distinguish between the following two distributions: a polynomial f/g with f and g sampled from distribution χ_k (assuming g is invertible over R_q) and a polynomial h sampled uniformly at random over R_q. These distributions were shown to be *statistically indistinguishable* for $\Phi_m\,(x) = x^n + 1$ when $\sigma_k = \omega\left(q^{1/2}\right)$[10]. Once the hardness of NTRU problem is proven (i.e., the public key cannot be distinguished from a uniformly distributed random polynomial), the RLWE assumption kicks in to prove semantic security of the encryption scheme. This logic was applied to show that the Stehlé-Steinfeld scheme defined by operations $\mathsf{KeyGen}, \mathsf{Enc}$, and Dec and constraint $\sigma_k = \omega\left(q^{1/2}\right)$ is IND-CPA secure [10].

Appendix A.3. Security Analysis and Constraints for NTRU-RLWE PRE Scheme

Though provably secure, the original Stehlé-Steinfeld scheme is impractical for proxy re-encryption or any homomorphic encryption scheme requiring at least one multiplication of two polynomials generated from the distribution χ_k. In this case, the correctness inequality for q would never hold as we would have $B_k^2 \propto \sigma_k^2 = \omega(q)$ on the right hand side of the expression, i.e., $q > \omega(q)$.

For practical reasons, the constraint $\sigma_k = \omega\left(q^{1/2}\right)$ was suggested to be replaced with a smaller value corresponding to the error distribution χ_e by arguing that the resulting Decisional Small Polynomial Ratio (DSPR) problem is secure against all known practical attacks [31]. This assumption was recently invalidated for some parameter ranges by two subfield lattice attacks [25, 50], which are able to reduce the ring dimension of the affected cryptosystems and solve the Shortest Vector Problem for n = 512 or lower.

Albrecht et al. [25] proposed a new practical constraint for σ_k based on the immunity of NTRU to subfield lattice attacks, conjecturing that the Stehlé-Steinfeld proof may be extended to this case:

$$\sigma_k > \left(\frac{2q}{n\pi e}\right)^{1/4}. \tag{A.3}$$

Our proxy re-encryption scheme, referred to as NTRU-RLWE PRE, uses this constraint. In contrast to the original Stehlé-Steinfeld scheme, our scheme supports ReKeyGen, ReEnc, and homomorphic indexing and multiplication operations.

To meet the RLWE security requirements of the encryption scheme, we use the inequality derived in [33], namely,

$$n \geq \frac{\log_2(q/\sigma_e)}{4\log_2(\delta)}. \tag{A.4}$$