March 2021

Efficient Post-Quantum and Compact Cryptographic Constructions for the Internet of Things

Rouzbeh Behnia

University of South Florida

Follow this and additional works at: https://digitalcommons.usf.edu/etd

Part of the Computer Sciences Commons

Scholar Commons Citation

This Dissertation is brought to you for free and open access by the USF Graduate Theses and Dissertations at Digital Commons @ University of South Florida. It has been accepted for inclusion in USF Tampa Graduate Theses and Dissertations by an authorized administrator of Digital Commons @ University of South Florida. For more information, please contact digitalcommons@usf.edu.
Efficient Post-Quantum and Compact Cryptographic Constructions for the Internet of Things

by

Rouzbeh Behnia

A dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy
Department of Computer Science and Engineering
College of Engineering
University of South Florida

Major Professor: Attila A. Yavuz, Ph.D.
Mehran Mozaffari Kermani, Ph.D.
Jay Ligatti, Ph.D.
Xinming (Simon) Ou, Ph.D.
Mike Rosulek, Ph.D.
Kaiqi Xiong, Ph.D.

Date of Approval:
March 25, 2021

Keywords: Digital signature, PEKS schemes, post-quantum cryptography

Copyright © 2021, Rouzbeh Behnia
Dedication

To my better half, Shima
Acknowledgments

First, I would like to extend my utmost gratitude to my advisor, Dr. Attila Yavuz for his continuous support, knowledge and guidance throughout this journey. He has always been available to answer my questions, give valuable feedback and wonderful discussions. A special thanks to my mentors at VISA Research Dr. Yilei Chen and Dr. Daniel Masny for the many fruitful discussions and insights during my internship.

I would like to thank my co-authors Dr. Elisa Bertino, Dr. Thang Hoang, Dr. Syed Rafiul Hussain, Dr. Yeongjin Jang, Muslum Ozgur Ozmen, Eamonn W. Postlethwaite, Dr. Mike Rosulek, Ankush Singla, and Dr. Tsz Hon Yuen for the many interesting discussions and knowledgeable insights.

I would like to express my appreciation to my colleges at Oregon State University, University of South Florida and VISA Research, Dr. Shashank Agrawal, Dr. Thang Hoang, Muslum Ozgur Ozmen, Dr. Peter Rindal, Efe Seyitoglu, Dr. Ni Trieu, Dr. Gaven Watson and all my friends at Oregon State University, University of South Florida and VISA Research.

I would like to thank my parents and my siblings for their continued love and support. Last but not least, I would like to thank my wife and best friend Shima for her continued love and support in every minute of this journey and for listening to me complain during the hard times.
# Table of Contents

List of Tables ....................................................................................... iv

List of Figures ....................................................................................... v

Abstract ................................................................................................ vi

Chapter 1 Introduction ........................................................................ 1
  1.1 Contributions ............................................................................. 2

Chapter 2 Preliminaries ....................................................................... 6
  2.1 Notations ...................................................................................... 6
  2.2 Definitions and Tools .................................................................. 6
    2.2.1 Digital Signatures ................................................................. 7
  2.3 Classical Hard Problems and Tools ............................................. 8
  2.4 Lattice-Based Problems and Tools ........................................... 9

Chapter 3 ARIS: Authentication for Real-Time IoT Systems ................ 14
  3.1 Introduction ............................................................................... 14
    3.1.1 Our Contributions ............................................................... 15
    3.1.2 Limitations .......................................................................... 16
  3.2 Related Work ............................................................................. 16
  3.3 Proposed Scheme ...................................................................... 18
  3.4 Security Analysis ...................................................................... 20
  3.5 Performance Evaluation ........................................................... 22
    3.5.1 Performance on Commodity Hardware ................................ 23
      3.5.1.1 Hardware Configurations ............................................ 23
      3.5.1.2 Software Libraries ...................................................... 23
      3.5.1.3 Parameter Choice ....................................................... 23
      3.5.1.4 Experimental Results ................................................ 24
    3.5.2 Performance on 8-bit AVR ................................................ 25
      3.5.2.1 Hardware Configurations ............................................ 25
      3.5.2.2 Software Libraries ...................................................... 25
      3.5.2.3 Parameter Choice ....................................................... 26
      3.5.2.4 Experimental Results ................................................ 26
      3.5.2.5 Energy Efficiency ....................................................... 27
  3.6 Conclusion .................................................................................. 27

Chapter 4 Compatible Certificateless and Identity-Based Cryptosystems for Heterogeneous IoT ........................................................................ 28
  4.1 Introduction ............................................................................... 28
    4.1.1 Our Contribution ................................................................. 30
  4.2 System and Security Model ....................................................... 32
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.2</td>
<td>Proposed PoW Protocol, (LP\omega)</td>
<td>89</td>
</tr>
<tr>
<td>6.3</td>
<td>Discussion</td>
<td>93</td>
</tr>
<tr>
<td>6.4</td>
<td>Related Work</td>
<td>94</td>
</tr>
<tr>
<td>7.1</td>
<td>Introduction</td>
<td>97</td>
</tr>
<tr>
<td>7.1.1</td>
<td>Research Gap</td>
<td>99</td>
</tr>
<tr>
<td>7.1.2</td>
<td>Our Contribution</td>
<td>100</td>
</tr>
<tr>
<td>7.1.3</td>
<td>Limitations</td>
<td>102</td>
</tr>
<tr>
<td>7.2</td>
<td>Background</td>
<td>103</td>
</tr>
<tr>
<td>7.2.1</td>
<td>Identity-Based Encryption</td>
<td>103</td>
</tr>
<tr>
<td>7.2.1.1</td>
<td>Abdalla et. al. IBE to PEKS transformation [2]</td>
<td>104</td>
</tr>
<tr>
<td>7.2.2</td>
<td>Public Key Encryption with Keyword Search</td>
<td>104</td>
</tr>
<tr>
<td>7.3</td>
<td>Proposed Schemes</td>
<td>107</td>
</tr>
<tr>
<td>7.3.1</td>
<td>PEKS Scheme from NTRU Lattices</td>
<td>107</td>
</tr>
<tr>
<td>7.3.1.1</td>
<td>Completeness and Consistency</td>
<td>109</td>
</tr>
<tr>
<td>7.3.1.2</td>
<td>Security Analysis</td>
<td>110</td>
</tr>
<tr>
<td>7.3.1.3</td>
<td>Discussion on Subfield Attacks</td>
<td>113</td>
</tr>
<tr>
<td>7.3.2</td>
<td>Lattice-Based PEKS Scheme in the Standard Model</td>
<td>113</td>
</tr>
<tr>
<td>7.3.2.1</td>
<td>Completeness and Consistency</td>
<td>115</td>
</tr>
<tr>
<td>7.3.2.2</td>
<td>Security Analysis</td>
<td>117</td>
</tr>
<tr>
<td>7.3.3</td>
<td>Secure Channel Requirement</td>
<td></td>
</tr>
<tr>
<td>7.3.4</td>
<td>Alternative NTRU-based Constructions</td>
<td>119</td>
</tr>
<tr>
<td>7.4</td>
<td>Performance Analysis and Comparison</td>
<td>120</td>
</tr>
<tr>
<td>7.4.1</td>
<td>Analytical Performance Comparison</td>
<td>120</td>
</tr>
<tr>
<td>7.4.1.1</td>
<td>Parameters</td>
<td>121</td>
</tr>
<tr>
<td>7.4.1.2</td>
<td>Computation</td>
<td>121</td>
</tr>
<tr>
<td>7.4.1.3</td>
<td>Storage and Communication</td>
<td>122</td>
</tr>
<tr>
<td>7.4.2</td>
<td>Evaluation Metrics and Experimental Setup</td>
<td>122</td>
</tr>
<tr>
<td>7.4.2.1</td>
<td>Evaluation Metrics</td>
<td>122</td>
</tr>
<tr>
<td>7.4.2.2</td>
<td>Software Libraries and Hardware Configurations</td>
<td>123</td>
</tr>
<tr>
<td>7.4.3</td>
<td>Performance Evaluation</td>
<td>124</td>
</tr>
<tr>
<td>7.4.4</td>
<td>Detailed Analysis and Cost Dissection</td>
<td>125</td>
</tr>
<tr>
<td>7.4.5</td>
<td>Discussion</td>
<td>128</td>
</tr>
<tr>
<td>7.5</td>
<td>Related Work</td>
<td>128</td>
</tr>
</tbody>
</table>

References .................................................................................. 130

Appendix A: Copyright Permissions .............................................. 147

About the Author ......................................................................... End Page
List of Tables

Table 3.1 Performance of ARIS and its counterparts on commodity hardware ............... 24
Table 3.2 Performance of ARIS and its counterparts on 8-bit AVR.......................... 25
Table 4.1 Analytical comparison of public key encryption schemes ......................... 47
Table 4.2 Analytical comparison of digital signature schemes............................... 47
Table 4.3 Analytical comparison of key exchange schemes.................................. 48
Table 4.4 Public key encryption schemes on commodity hardware.......................... 48
Table 4.5 Digital signature schemes on commodity hardware.................................. 49
Table 4.6 Key exchange schemes on commodity hardware.................................. 49
Table 4.7 Public key encryption schemes on 8-bit AVR processor........................... 50
Table 4.8 Digital signature schemes on 8-bit AVR processor................................ 50
Table 4.9 Key exchange schemes on 8-bit AVR processor .................................. 51
Table 5.1 Parameter selection of TACHYON.................................................. 76
Table 5.2 Experimental performance comparison of TACHYON with its counterparts........ 80
Table 7.1 Parameter our schemes and their counterparts ................................... 120
Table 7.2 Micro benchmark of our schemes and their counterparts ........................ 120
### List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 3.1</td>
<td>Energy consumption (mJ) for signing on microcontroller</td>
<td>17</td>
</tr>
<tr>
<td>Figure 3.2</td>
<td>Energy consumption (mJ) for verification on microcontroller</td>
<td>17</td>
</tr>
<tr>
<td>Figure 4.1</td>
<td>Proposed IDB and CL cryptosystems and alternatives (high-level)</td>
<td>29</td>
</tr>
<tr>
<td>Figure 7.1</td>
<td>PEKS scheme in secure email system</td>
<td>98</td>
</tr>
<tr>
<td>Figure 7.2</td>
<td>End-to-end search time of NTRU-PEKS for $\kappa = 192$</td>
<td>125</td>
</tr>
<tr>
<td>Figure 7.3</td>
<td>End-to-end search time of LWE-PEKS for $\kappa = 192$</td>
<td>126</td>
</tr>
<tr>
<td>Figure 7.4</td>
<td>End-to-end search time comparison for $\kappa = 192$</td>
<td>126</td>
</tr>
</tbody>
</table>
Abstract

IoT systems often rely on low-end devices to send measurements to other parties and depending on the setting, unauthorized alteration and/or privacy violation of these measures can have catastrophic consequences (e.g., embedded medical sensors). Therefore, providing efficient authentication, integrity, and confidentiality in these settings is vital. While conventional cryptographic measures (e.g., ECDSA) can be used to meet these security requirements, despite their elegant design, they are often too computationally expensive for low-end devices. This is further exacerbated when security against quantum computers is taken into the account.

In this dissertation, we propose a series of new efficient conventional and post-quantum cryptographic schemes to meet the stringent requirement of such IoT systems. In the line of proposing efficient authentication schemes, we propose two signature schemes. Our first signature scheme is based on conventional cryptographic problems and utilizes the message encoding with cover-free families and special property of ECDLP-based functions to achieve significant performance gain as compared to its counterparts. The second scheme is based on post-quantum primitives and is achieved by extending one-time signatures to (polynomially bounded) many-time signatures, using the additively homomorphic properties of generalized compact knapsack functions. The new scheme achieves the lowest end-to-end delay among its counterparts which makes it suitable for low-end devices. As a step toward a fully post-quantum blockchain, we propose a Proof of Work (PoW) protocol that minimizes the advantage of a quantum miner. Our new protocol is based on the Hermite Shortest Vector Problem (Hermite-SVP) in the Euclidean norm and allows for a fast verify algorithm. To alleviate the hurdle of certificate communication and verification for low-end devices.
devices, we then present an identity-based and certificateless cryptosystems that are created using special key generation algorithms that harness the additive homomorphic property of the exponents to enable the users to incorporate their private keys into the one provided by the trusted third party without falsifying it. The new schemes achieve better computation efficiency and comparable communication efficiency as compared to their identity-based and certificateless counterparts. Lastly, with the aim of proposing efficient and highly-secure measures for secure remote data storage, we propose two lattice-based public key searchable encryption schemes with post-quantum security. To our knowledge, our schemes are the first instances of such schemes based on lattices that provide a post-quantum promise. Our first variant is based on NTRU lattices and provides a significant performance advantage and better end-to-end delay as compared to its existing counterparts. The second scheme, based on the LWE problem in the standard model, provides a better security as compared to its counterparts with a cost of an inferior performance. All of the proposed schemes are proven secure via rigorous security proofs and are implemented and open-sourced to allow for public testing and verification.
Chapter 1: Introduction

Internet of Things (IoT) is a heterogeneous system consisting of a large number of interconnected sensors, smart devices, transceivers, microcomputers, etc. Such systems often rely on real-time communication to provide the intended functionality and can be targeted by adversarial attacks to target the authentication, integrity, and/or privacy of the data being transmitted/stored. Fortunately, there exist a wide array of cryptographic schemes, designed to effectively mitigate/thwart these attacks.

These cryptographic schemes can be mainly divided into symmetric key based or public key based systems. Symmetric key based schemes offer highly efficient and secure solutions, however, they might not be ideal for adoption in some IoT setting due to the following: (i) Shared key computation and distribution: Symmetric key based schemes require a key agreement protocol to compute a shared secret key before initiating a secure communication. While it might be feasible to pre-load these shared keys on all the devices in some applications, it would be quite difficult for systems where moving sensors/devices need to communicate with a plethora of new devices in a real-time manner (e.g., aerial drone networks, vehicular network, etc.). (ii) Storage of shared keys: For large IoT networks with thousands or even millions of devices, the storage of the pre-computed/pre-shared keys might not be possible on low-end devices due to their limited storage. For instance, an ATmega2560 has 256 KB of the flash memory of which 8KB is used by the bootloader. (iii) Lack of public verifiability and non-repudiation in authentication schemes: While there exist many symmetric key-based notions to provide authentication, they fail to provide non-repudiation
and public verifiability. This is mainly because such schemes require the signing key to verify the authenticity of the authenticating tokens.

Public key based schemes are designed to address the above shortcomings and provide more advanced security properties. However, despite their elegance, these systems (e.g., [15, 52]) are often too expensive for adoption in some IoT applications that consist of low-end devices (often battery powered) and/or are delay-aware [68]. Furthermore, given the eventual debut of quantum computers poses an imminent threat to classical hard problems, which most, if not all, of the existing public key cryptosystems rely on, the plans for migration to post-quantum secure systems has already begun by standardization and government bodies (e.g., NIST, NSA, etc.). However, as compared to their conventional counterparts, these post-quantum secure solutions often incur more computation, storage, and/or communication overhead which makes their adoption on the aforementioned IoT setting even more challenging.

1.1 Contributions

Our goal is to bridge this research gap by devising new public key based schemes that can meet the scalability and stringent performance requirements of low-end IoT infrastructures. Therefore, in this dissertation, we present a series of practical conventional and post-quantum secure cryptographic measures (e.g., digital signatures, certificate-free cryptosystems, public key searchable encryption schemes, etc.) based on well-studied assumptions to be deployed in applications such as smart grid systems, smart implantable medical devices, Internet of drones, and secure cloud storage. The new schemes are designed with either a new/improved theoretical foundation or systems design that can help to bridge the gap between functionality, efficiency, and security. We highlight the contributions of this dissertation in more detail in the following.
1. **Highly efficient digital signature scheme:** We propose a new signature scheme called **ARIS** [30] that pushes the limits of the existing digital signatures, with nearly $2 \times$ faster verification and $33\%$ faster signing, compared to its fastest counterpart [79]. This efficiency gain is achieved by harnessing message encoding with cover-free families and special elliptic curve based one-way function. This significant computational advantages come with a larger storage requirement, which is a highly favorable trade-off for some critical delay-aware applications. We prove the security of **ARIS** under the hardness of the elliptic curve discrete logarithm problem (in the random oracle model) and provide an open-sourced implementation of **ARIS** on commodity hardware and 8-bit AVR microcontroller which confirmed the significant performance gain of **ARIS**.

2. **Compatible certificate-free cryptosystems for the IoT:** Certificate-free cryptosystems, like identity-based [53] and certificateless systems [8], lift the burden of certificate (chain) communication and verification which might be too costly for some IoT systems. We propose new identity-based and certificateless cryptosystems [33] that aside from being more efficient than their counterparts, provide compatibility to enable users from different domains (identity-based or certificateless) to communicate seamlessly. This feature is beneficial for some heterogeneous IoT settings (e.g., aerial drones [162]), where different levels of trust/control is assumed on the trusted third party. The idea behind our constructions is to design special key generation algorithms that use the exponent product of powers property and cover-free functions (similar to [179]) to enable users to incorporate their private keys into the one provided by the trusted third party without falsifying it. We prove the security of our schemes (in the random oracle model) and open-source our cryptographic framework for public testing and adoption.
3. Fast post-quantum signatures from compact knapsack: We introduce a simple, yet efficient digital signature scheme which offers post-quantum security promise. The new scheme, named TACHYON, is based on a novel approach for extending one-time hash-based signatures to (polynomially bounded) many-time signatures, using the additively homomorphic properties of generalized compact knapsack (GCK) functions. Our design permits TACHYON to achieve several key properties. First, its signing and verification algorithms are the fastest among its current counterparts with a higher level of security. This allows TACHYON to achieve the lowest end-to-end delay among its counterparts, while also making it suitable for resource-limited signers. Second, its private keys can be as small as $\kappa$ bits, where $\kappa$ is the desired security level. Third, unlike most of its lattice-based counterparts, TACHYON does not require any Gaussian sampling during signing, and therefore, is free from side-channel attacks targeting this process. We prove the security of TACHYON based on the one-wayness of GCK function family.

4. Post-quantum proof-of-work for post-quantum blockchain: Proof of Work (PoW) protocols, originally proposed to circumvent DoS and email spam attacks, are now at the heart of the majority of recent cryptocurrencies. Given the potential application of TACHYON to provide efficient authentication in post-quantum blockchains, in a step toward a fully post-quantum secure blockchain, we propose a new PoW protocol. Current popular PoW protocols are based on hash puzzles. By considering the hash as a random function, and fixing a priori a sufficiently large search space, Grover’s search algorithm [113] gives an asymptotic quadratic advantage to quantum machines over classical ones. In this work, we propose a PoW protocol for which quantum machines have a smaller asymptotic advantage. Specifically, for a lattice of rank $n$ sampled from a particular class, our protocol provides as the PoW an instance of the Hermite Shortest Vector Problem (Hermite-SVP). Asymptotically, the best known classical and quantum algorithms that directly solve SVP type problems are heuristic lattice sieves,
which run in time $2^{0.292n+o(n)}$ and $2^{0.265n+o(n)}$ respectively. We discuss recent advances in SVP type problem solvers and give examples of where the impetus provided by a lattice based PoW would help explore often complex optimization spaces.

5. **Lattice-based public key encryption with keyword search (PEKS) Schemes**: PEKS schemes [52] aim in mitigating the impacts of data privacy versus utilization dilemma by allowing *any user in the system* to send encrypted files to the server to be searched by a receiver who poses the private key. The existing PEKS schemes introduce a high end-to-end delay that may hinder their adoption in practice. In this work, we propose two novel lattice-based PEKS schemes [28, 32] that offer a high computational efficiency along with better security compared to their counterparts [52, 214]. Our NTRU-PEKS scheme achieves $18 \times$ lower end-to-end delay than its most efficient counterpart. This is due to a highly efficient Test algorithm that runs linear to the number of keyword-file pairs. Our LWE-PEKS offers provable security in the standard model with a reduction to the worst-case lattice problems with a cost of a higher end-to-end delay and parameter sizes. We fully implemented our NTRU-PEKS scheme and benchmarked its performance as deployed on Amazon Web Services cloud infrastructures.
Chapter 2: Preliminaries

In this chapter, we present our notations, definitions, and tools that will be used throughout this dissertation.

2.1 Notations

$x \overset{\$}{\leftarrow} S$ denotes randomly selecting $x$ from a set $S$. We denote scalars as small letters (e.g., $x$) and points on elliptic curves (EC) as capital letters (e.g., $P$). We denote vectors as bold letters (i.e., $v$), while matrices are denoted as bold capital letters (i.e., $V$). $x \overset{\$}{\leftarrow} S$ denotes that $x$ is being randomly selected from set $S$. $|x|$ denotes the bit length of a number $x$, i.e., $|x| = \log_2 x$. EC scalar multiplication is denoted as $xP$, and all EC operations use an additive notation. For $n \in \mathbb{N}^+$ let $[n] = \{1, \ldots, n\}$. For a finite set $S$, let $x \leftarrow U(S)$ denote a uniform sample. Let $m(n)$ represent the cost of multiplying two $n$ bit numbers. Let $\| \cdot \|$ represent the Euclidean norm.

$\mathcal{A}^{O_1 \ldots O_n}(\cdot)$ denotes algorithm $\mathcal{A}$ is provided with access to oracles $O_1 \ldots O_n$. For a vector $w = (w_1, \ldots, w_n)$ we define $\|w\|_\infty = \max\{|w_i| : i = 1, \ldots, n\}$. $\lfloor x \rfloor$ rounds $x$ to the closest integer. $x \overset{\Delta}{=} y$ means $x$ is defined as $y$. The function $\gcd(x, y)$ returns the greatest common divisor of values $x$ and $y$.

2.2 Definitions and Tools

In this section, we present the definitions that will be referred to throughout this dissertation.
2.2.1 Digital Signatures

**Definition 1.** A digital signature scheme is a tuple of three algorithms $SGN = (Kg, Sig, Ver)$ defined as follows.

- $(sk, pk) \leftarrow SGN.Kg(1^\kappa)$: Given the security parameter $\kappa$, it outputs a private/public key pair $(sk, pk)$.

- $\sigma \leftarrow SGN.Sig(M, sk)$: Given a message $M$ and private key $sk$, it outputs a signature $\sigma$.

- $\{0,1\} \leftarrow SGN.Ver(M, \sigma, pk)$: Given a message-signature pair $(M, \sigma)$, and $pk$, it outputs a decision bit $d \in \{0,1\}$.

We say that $SGN$ is correct if for all $(sk, pk) \leftarrow SGN.Kg(1^\kappa)$, $SGN.Ver(M, SGN.Sig(M, sk), pk) = 1$ holds.

In the following definition, we define the security of signature schemes based on the methodology proposed in [40]. After the initialization phase i.e., $SGN.Kg(\cdot)$, The adversary $A$ is given access to the signature generation oracle. $A$ wins, if it outputs a valid message-signature pair (that was not previously outputted from the sign oracle) after making polynomially-bounded number of queries.

**Definition 2.** Existential Unforgeability under Chosen Message Attack (EU-CMA) experiment $\text{Expt}_{SGN}^{EU-CMA}$ is defined as follows.

- $(sk, pk) \leftarrow SGN.Kg(1^\kappa)$

- $(m^*, \sigma^*) \leftarrow A^{SGN.Sig(\cdot)}(pk)$

- If $1 \leftarrow SGN.Ver(m^*, \sigma^*, pk)$ and $m^*$ was not queried to $SGN.Sig(\cdot)$, return 1, else, return 0.

The EMU-CMA advantage of $A$ is defined as $Adv_{SGN}^{EU-CMA} = \Pr[\text{Expt}_{SGN}^{EU-CMA} = 1]$. 
Generalized Forking Lemma (GFL) [38] is a commonly used technique in the security proof of various well-studied digital signature schemes (e.g., Schnorr [183]). Intuitively, GFL states that if an adversary can successfully generate a forgery, then it is possible to rewind the adversary, choose new random oracle responses after a certain point, and the adversary will still be able to generate a forgery with polynomially-related probability.

**Lemma 1.** (General Forking Lemma [38]) Fix an integer $q_F \geq 1$ and a set $H$ of size $h_F \geq 2$. Let $A$ be a randomized algorithm that returns a pair $(J, \sigma)$ where $J \in \{0, \ldots, h_{qF}\}$ and $\sigma$ is the side output, on the input of $(x, h_1, \ldots, h_{qF})$. For $\mathcal{IG}$ as a randomized input generator, the accepting probability of $A$ ($ACC$) is defined as the probability that $J \geq 1$ in $x \xleftarrow{\$} \mathcal{IG}; (h_1, \ldots, h_{qF}) \xleftarrow{\$} H; (J, \sigma) \xleftarrow{\$} A(x, h_1, \ldots, h_{qF})$.

The forking algorithm $\text{Fork}_A$ associated with $A$ is a randomized algorithm that behaves as in Algorithm 1. For $\text{FRK} = \Pr[b = 1 : x \xleftarrow{\$} \mathcal{IG}; (b, \sigma, \sigma') \xleftarrow{\$} \text{Fork}_A(x)]$, then $\text{FRK} \geq ACC \cdot \left(\frac{ACC}{q_F} - \frac{1}{h_F}\right)$.

**Algorithm 1** Forking algorithm $\text{Fork}_A$ for the forking lemma.

1: Pick coins $\rho$ for $A$ at random.
2: $(h_1, \ldots, h_{qF}) \xleftarrow{\$} H$
3: $(I, \sigma) \leftarrow A(x, h_1, \ldots, h_{qF}; \rho)$
4: If $I = 0$ then return $(0, 0, 0)$
5: $(h'_1, \ldots, h'_{qF}) \xleftarrow{\$} H$
6: $(I', \sigma') \leftarrow A(x, h_1, \ldots, h_{I-1}, h'_I, \ldots, h_{qF}; \rho)$
7: If $(I = I'$ and $h_I \neq h'_I$, return $(1, \sigma, \sigma')$
8: Else, return $(0, 0, 0)$

### 2.3 Classical Hard Problems and Tools

FourQ [78] is a special EC that is defined by the complete twisted Edwards equation $\mathcal{E}/F_{p^2}$: $-x^2 + y^2 = 1 + dx^2y^2$. FourQ is known to be one of the fastest elliptic curves that admits 128-bit security level [78]. Moreover, with extended twisted Edwards coordinates, FourQ offers the fastest EC addition algorithms [78], that is extensively used in our optimizations. All of our schemes are
realized on FourQ.

**Definition 3.** Given points \( P, Q \in E(\mathbb{F}_p) \), the Elliptic Curve Discrete Logarithm Problem (ECDLP) asks to find \( a \), if it exists, such that \( aP \mod p = Q \).

**Definition 4.** Given \( P, aP, bP \in E(\mathbb{F}_p) \), the Computational Diffie-Hellman Problem (CDHP) asks to compute \( abP \).

### 2.4 Lattice-Based Problems and Tools

An \( n \) dimensional lattice \( \Lambda \) of rank \( k \leq n \) is a discrete additive subgroup of \( \mathbb{R}^n \). Given \( k \) linearly independent basis vectors \( \{b_1, \ldots, b_k\} \subset \mathbb{R}^n \), the lattice generated by \( \mathbf{B} \), i.e. their concatenation as column vectors, is

\[
\Lambda(\mathbf{B}) = \Lambda(b_1, \ldots, b_k) = \left\{ \sum_{i=1}^{k} x_i \cdot b_i : x_i \in \mathbb{Z} \right\}.
\]

The volume of \( \Lambda \) is defined as

\[
\text{Vol}(\Lambda) = \sqrt{\det(\mathbf{B}^t \mathbf{B})}
\]

for any basis \( \mathbf{B} \) of \( \Lambda \) (i.e. volume is an invariant of the lattice, and independent of the choice of basis). We will consider only full rank lattices, where \( n = k \) and \( \text{Vol}(\Lambda) = \det(\mathbf{B}) \).

Ajtai [5] introduced the Short Integer Solution (SIS) problem and demonstrated the connection between average-case SIS problem and worst-case problems over lattices. Hoffstein et al. [119] proposed a very efficient public key encryption scheme based on NTRU lattices. Regev [177]
introduced the Learning with Error (LWE) problem. The SIS and LWE problems have been used as the building blocks of many lattice-based schemes.

NTRU encryption works over rings of polynomials $\mathcal{R} = \mathbb{Z}[x]/(x^N+1)$ and $\mathcal{R}' = \mathbb{Q}[x]/(x^N+1)$ which are parametrized with $N$ as a power-of-two integer. $(x^N+1)$ is irreducible, therefore, $\mathcal{R}'$ is a cyclotomic field. For $f = \sum_{i=0}^{N-1} f_i x^i$ and $g = \sum_{i=0}^{N-1} g_i x^i$ as polynomials in $\mathbb{Q}[x]$, $fg$ denotes polynomial multiplication in $\mathbb{Q}[x]$ while $f \ast g \triangleq fg \mod (x^N+1)$ is referred to as convolution product. For an $N$-dimension anti-circulant matrix $\mathcal{A}_N$ we have $\mathcal{A}_N(f) + \mathcal{A}_N(g) = \mathcal{A}_N(f + g)$, and $\mathcal{A}_N(f) \times \mathcal{A}_N(g) = (f \ast g)$.

**Definition 5.** For prime integer $q$ and $f, g \in \mathcal{R}$, $h = g \ast f^{-1} \mod q$, the NTRU lattice with $h$ and $q$ is $\Lambda_{h,q} = \{(u,v) \in \mathcal{R}^2 : u + v \ast h = 0 \mod q\}$. $\Lambda_{h,q}$ is a full-rank lattice generated by $\mathcal{A}_{h,q} = \begin{bmatrix} \mathcal{A}_N(h) & I_N \\ qI_N & 0_N \end{bmatrix}$, where $I$ is an identity matrix.

Note that one can generate this basis using a single polynomial $h \in \mathcal{R}_q$. However, the lattice generated from $\mathcal{A}_{h,q}$ has a large orthogonal defect which results in the inefficiency of standard lattice operations. As proposed by [118], another basis (which is much more orthogonal) can be efficiently [88] generated by selecting $F, G \in \mathcal{R}$ and computing $f \ast G - g \ast F = q$. The new base $B_{f,g} = \begin{bmatrix} \mathcal{A}(g) & -\mathcal{A}(f) \\ \mathcal{A}(G) & -\mathcal{A}(F) \end{bmatrix}$ generates the same lattice $\Lambda_{h,q}$.

**Definition 6.** The minimum distance of a lattice $\Lambda$ is $\lambda_1(\Lambda) = \min \{\|v\| : v \in \Lambda \setminus \{0\}\}$ . A solution to $\gamma$-approx-SVP for $\gamma \geq 1$ is a vector $v \in \Lambda \setminus \{0\}$ such that $\|v\| \leq \gamma \cdot \lambda_1(\Lambda)$.

An immediate corollary of Minkowski’s theorem, in the Euclidean norm, proves that

$$\lambda_1(\Lambda) \leq \sqrt{n} \cdot \text{Vol}(\Lambda)^{1/n}.$$
The Gaussian heuristic estimates the number of lattice points of a lattice \( \Lambda \) contained in a measurable set \( S \) as \( \text{Vol}(S)/\text{Vol}(\Lambda) \). When applied to a hypersphere it gives the following estimate for \( \lambda_1(\Lambda) \).

**Definition 7.** Let the Gaussian heuristic estimate, \( \text{gh}(\Lambda) \), for \( \lambda_1(\Lambda) \) be given using the Gamma function, as \( \text{gh}(\Lambda) = \frac{\Gamma(n/2+1)^{1/n}}{\sqrt{\pi}} \cdot \text{Vol}(\Lambda)^{1/n} \).

We note that the above is a heuristic for the length of the shortest non zero vectors in a lattice, and the existence of a vector slightly larger than this heuristic will be important for our construction. There are many asymptotic and experimental works that determine the usefulness of the Gaussian heuristic in different settings. For a theoretical introduction, see e.g. [72, Section 3.1.2], for experimental evidence that it is accurate for \( n \geq 50 \) see [72, Section 3.1.3], and for an asymptotic statement see e.g. [140, Thm 4]. More practically, Blichfeldt’s inequality [50] tells us that for \( n > 24 \), any lattice \( \Lambda \) has \( \lambda_1(\Lambda) \leq \sqrt{2} \cdot \text{gh}(\Lambda) \). We will only consider \( n > 24 \) in our work in Chapter 6.

**Definition 8.** The \( \alpha \)-Hermite-SVP, or \( \alpha \)-HSVP problem is, given a lattice \( \Lambda \), to find a vector \( \mathbf{v} \in \Lambda \setminus \{0\} \) such that \( \|\mathbf{v}\| \leq \alpha \cdot \text{Vol}(\Lambda)^{1/n} \).

Instances of Hermite-SVP are given as a lattice basis, which much be somehow sampled. We generate Goldstein–Mayer lattices [109] as \( \Lambda(\mathbf{B}) \) for

\[
\mathbf{B} = \begin{pmatrix}
p & x_2 & \cdots & x_n \\
0 & 1 & \cdots & 0 \\
\vdots & & \ddots & 0 \\
0 & 0 & 0 & 1
\end{pmatrix}
\] (2.1)

where \( p \) is a large prime and \( x_i \leftarrow \mathcal{U}(\{0\} \cup [p-1]) \) are i.i.d. uniform. These lattices have \( \text{Vol}(\Lambda) = p \) and provide a way to sample "uniformly" from all lattices of this volume [72, Section 2.3]. For
example, the Darmstadt SVP Challenge\footnote{https://www.latticechallenge.org/svp-challenge/} uses $\log_2 p \approx 10n$ and sets

$$\alpha = 1.05 \cdot \Gamma(n/2 + 1)^{1/n}/\sqrt{\pi}.$$

This $\alpha$ is such that a solution to $\alpha$-HSVP has length at most $1.05 \cdot \text{gh}(\Lambda)$, and is therefore a constant factor smaller than

$$\alpha' = \sqrt{2} \cdot \Gamma(n/2 + 1)^{1/n}/\sqrt{\pi}$$

that guarantees a solution to $\alpha'$-HSVP. However, we expect $1.05^n$ lattice vectors of length at most $1.05 \cdot \text{gh}(\Lambda)$ for $n \gtrsim 50$\footnote{[72, Section 3.1]}, and the probability of such a short vector not existing to be negligible.

**Definition 9.** (Statistical Distance [4]) Given two random variables $X$ and $Y$ taking values in a finite set $S$, the statistical distance is defined as:

$$\Delta(X, Y) = \frac{1}{2} \sum_{s \in S} |\Pr[X = s] - \Pr[Y = s]|$$

$X$ is said to be $\delta$–uniform over $S$ if $\Delta(X, Y) \leq \delta$.

Using Gaussian sampling, Gentry et al. [104] proposed a technique to use a short basis as a trapdoor without disclosing any information about the short basis and prevent attacks similar as in [159].

**Definition 10.** An $N$-dimensional Gaussian function $\rho_{\sigma, c} : \mathbb{R} \to (0, 1]$ is defined as $\rho_{\sigma, c}(x) \triangleq \exp(-\frac{\|x-c\|^2}{2\sigma^2})$. Given a lattice $\Lambda \subset \mathbb{R}^n$, the discrete Gaussian distribution over $\Lambda$ is $D_{\Lambda, \sigma, c}(x) = \frac{\rho_{\sigma, c}(x)}{\rho_{\sigma, c}(\Lambda)}$ for all $x \in \Lambda$.\footnote{[159]}
In [177], Regev has shown that for a certain error distribution $\chi$, denoted $\bar{\Psi}_\alpha$, the LWE problem is as hard as the worst-case SIVP and GapSVP under quantum reduction.

**Definition 11.** (Distribution of $\bar{\Psi}_\alpha$ [169, 177]) For $\alpha \in (0, 1)$ and a prime $q$, $\bar{\Psi}_\alpha$ denotes the distribution over $\mathbb{Z}_q$ of the random variable $\lfloor qX \rfloor \mod q$ is a normal random variable with mean 0 and the standard deviation $\frac{\alpha}{\sqrt{2\pi}}$.

Our scheme in Chapter 5 uses the generalized compact knapsack (GCK) function family, introduced by Micciancio [154].

**Definition 12 ([154]).** For a ring $\mathcal{R}$, and a small integer $\mu > 1$, the generalized compact knapsack function family is the set of functions of the form $F_A : \mathcal{R}^\mu \to \mathcal{R}$, where:

$$F_A(b_1, \ldots, b_\mu) = \sum_{i=1}^\mu b_i \cdot a_i$$

An instance of this family is specified by $\mu$ fixed elements $A = (a_1, \ldots, a_\mu) \in \mathcal{R}^\mu$. These elements are to be chosen randomly and independently. The inputs $b_1, \ldots, b_\mu$ are polynomials over $\mathcal{R}$ where $\|b_i\|_\infty \leq \beta$ for $i \in \{1, \ldots, \mu\}$ and some positive integer $\beta$.

For the detailed security analysis of GCK function, we refer an interested reader to [147, 154, 155, 170].
Chapter 3: Authentication for Real-Time IoT Systems

3.1 Introduction

IoT systems often need authentication for applications that need to verify a large volume of incoming transactions or commands. While symmetric key primitives (e.g., HMAC) can provide very fast authentication, they fail to offer non-repudiation which is often vital for these applications. For instance, Visa handles millions of transactions every day [192]. Each transaction corresponds to multiple authentications of the user’s request and card information on merchant’s side, payment gateway and credit card issuer [164]. Therefore, creating more efficient solutions can significantly reduce the overall authentication overhead of such systems that results in substantial financial gains.

The need for efficient authentication becomes even more imperative for applications in which IoT devices must operate in safety-critical settings and/or with battery limitations. For instance, battery-powered aerial drones [209] might communicate and authenticate streams of commands and measurements with an operation center in a short period of time. A fast and energy-efficient authentication can improve the flight and response time of such aerial drones [163]. Other IoT applications such as smart grid systems and IoT , which involve battery-powered sensors, will also benefit from fast and energy-efficient digital signatures which minimize the authentication delay/overhead and improve the operation time of the sensors [197]. In vehicular networks, safety significantly hinges on the end-to-end delay [1], and therefore attaining a signature scheme with the lowest end-to-end delay is always desired. Additionally, efficient signature schemes have been shown to be useful in digital forensic and security of log [185].

\[\text{This chapter was published in [27]. Permission is included in Appendix A.}\]
3.1.1 Our Contributions

In this chapter, we propose a new efficient signature scheme called ARIS.

ARIS makes use of an Elliptic Curve Discrete Logarithm Problem (ECDLP) based one-way function and exploits the homomorphic properties of such functions to (i) linearly add the private key elements to attain a shorter signature and (ii) mask this addition with a one-time randomness $r$ to achieve a (polynomially-bounded) multiple-time signature scheme. We outline the main properties of ARIS as below.

- **Fast Verification:** ARIS provides the fastest signature verification among its counterparts. More specifically, ARIS pushes the limits of elliptic curve (EC) based signature schemes by providing nearly $2 \times$ faster verification as compared to its fastest counterpart [79].

- **Fast Signing:** The signature generation of ARIS avoids expensive computations such as fixed-base scalar multiplication. Therefore, ARIS achieves 33% faster signing as compared to its fastest counterpart [79].

- **Low End-to-End Delay:** Due to having the fastest signature generation and verification algorithms, ARIS achieves nearly 40% lower end-to-end delay, as compared to its fastest counterpart [79]. This might encourage the potential adoption of ARIS for applications that require delay-aware authentication.

- **Energy Efficiency:** By avoiding any computationally expensive operation in the signing and verification algorithms, ARIS achieves the lowest energy consumption as compared to its state-of-the-art efficient counterparts. Specifically, as shown in Figure 3.1, the verification algorithm in ARIS attains 40% lower energy consumption as compared to its most energy
efficient counterpart. This makes ARIS potentially suitable for IoT applications wherein the battery-powered devices authenticate telemetry and commands (e.g., aerial drones).

- **Tunable Parameters**: ARIS enjoys from a highly tunable set of parameters. This allows ARIS to be instantiated with different properties for different applications. For instance, the parameters set that we considered for our implementation on AVR microcontroller enjoys from a smaller public key and private key pair, and if the same scheme is implemented on commodity hardware, it can enjoy from a faster signature generation (2× faster than the scheme in [79]) by incurring a few microseconds on the verification algorithm.

3.1.2 Limitations

All of the desired properties and efficiency gains in ARIS come with the cost of larger key sizes. For instance, in the verification efficient instantiation of ARIS (as in Table 3.1), which has the largest key sizes, the size of the public key and private key could be as large as 32KB. However, this can be decreased to 16KB and 8KB for the private key and public key sizes (respectively) while still maintaining the fastest signature generation and verification algorithms among its counterparts. We have shown that even with these parameters sizes, ARIS can be implemented on 8-bit AVR while enjoying from the most computation and energy efficient algorithms as shown in Figure 3.1, Figure 3.2 and Table 3.2.

3.2 Related Work

One-time signatures (e.g. HORS [179]) have been proposed to offer fast signing and verification. Following HORS, many schemes with different performance and security trade-offs such as time valid one-time signatures (i.e., TV-HORS [206]) have been proposed. However, these schemes suffer from security and performance penalties incurred due to the need for time-synchronization.
and their low tolerance for packet loss. Multiple-time hash-based signatures (e.g., XMSS \[66\]) utilize Merkle-Tree and can sign multiple messages by keeping the signer's state. Recently, stateless variations (e.g., SPHINCS \[46\]) have been proposed, however such schemes suffer from large signatures (\(\approx 41\) KB) and slow signing algorithms.

Recently, a polynomially-bounded multiple-time signature scheme based on HORS design is proposed \[31\]. The scheme utilizes the additive homomorphic property of the underlying one-way function to obtain fast signatures where the signer only aggregates private key components during the online phase. However, despite its efficiency, it cannot meet the stringent delay requirement of some IoT applications. Another proposed scheme called CEDA \[161\] exploits the aggregatable property of RSA-based one-way permutation functions and message encoding (as proposed in \[179\]) to attain efficient signing. However, the large parameter sizes not only incur very large public keys but also make the exponentiations that takes place during signature generation and verification quite costly. Therefore CEDA, while being among the most efficient schemes, does not surpass the latest implementations of signatures on fast elliptic curves.

In the line of proposing fast elliptic curves, Renes et al. \[44\] presented an efficient instantiations of the scheme in \[45\] based on Kummer surface that shows significant performance gains as compared
to its base scheme [45]. In 2016, Costello et al. [79] proposed a new implementation of [45] based on another elliptic curve called FourQ which shows to even outperform the implementation in [44].

3.3 Proposed Scheme

Before we propose our scheme, we define two Pseudo Random Functions $\text{PRF}_1 : \{0,1\}^* \rightarrow \mathbb{Z}_q$ and $\text{PRF}_2 : \{0,1\}^* \rightarrow \{0,1\}^\kappa$ and three hash function $H_1 : \{0,1\}^* \times \mathbb{Z}_q \rightarrow \{0,1\}^{l_1}$, $H_2 : E(\mathbb{F}_p) \rightarrow \{0,1\}^{l_2}$, and $H_3 : \{0,1\}^* \times \{0,1\}^{l_2} \rightarrow \{0,1\}^{l_1}$ for some integers $l_1$ and $l_2$, to be defined in §3.5.

ARIS leverages the homomorphic property of its underlying ECDLP-based one-way function, which is due to the exponent product of powers property, to achieve (polynomially-bounded) multiple-time signatures from the one-time signature scheme proposed in [180], with more compact signatures. More specifically, in ARIS, the private key consists of $t$ randomly generated values $x_i$ (generated using a $\kappa$ bit seed $z$) and the corresponding public key consists of all $Y_i \leftarrow x_i P$ for $i \in \{1, \ldots, t\}$.

To sign a message, the signer obtains $k$ indexes $(i_1, \ldots, i_k)$ by hashing the message (and a random input), uses the indexes $(i_1, \ldots, i_k)$ to retrieve the corresponding private key elements (i.e., $x_{i_j}$ where $j \in \{1, \ldots, k\}$) and sums them along with a one-time randomness $r$. The signature consists of $s$ and $h$, which is obtained by applying the hash function $H_2(\cdot)$ on $R$, that is computed as the output of applying the one-way function on the one-time randomness $r$.

Verification takes place by computing the summation of the corresponding public key elements (i.e., $Y_{i_j}$) and their subtraction from the output of the ECDLP-based one-way function applied on $s$. The verifier outputs valid if the subtraction yields the same value of $R$ as computed in the signature generation. Additionally, ARIS uses the BPV method in [63] to convert an EC scalar multiplication to only $k$ (where $k = 18$ or $k = 28$ for our proposed parameter sets) EC point additions with the cost of storing a small, constant-size table.
Our scheme consists of the following algorithms.

\((sk, pk) \leftarrow \text{ARIS.Kg}(1^\kappa)\): Given the security parameter \(\kappa\), this algorithm selects parameters \((t, k)\) such that \(\left(\frac{t}{k}\right) \geq 2^\kappa\) and \(z \leftarrow \mathbb{Z}_q\) and works as follows.

- Compute \(x_i \leftarrow \text{PRF}_1(z, i)\) and \(Y_i \leftarrow x_i P\) for \(i \in \{1, \ldots, t\}\) and set \(Y \leftarrow \{Y_i\}_{i=1}^t\).
- Compute \(r_i \leftarrow \text{PRF}_2(z, i)\) and \(R_i \leftarrow r_i P\) for \(i \in \{1, \ldots, t\}\) and set \(R \leftarrow \{R_i\}_{i=1}^t\).
- Output \(pk \leftarrow Y\) and \(sk \leftarrow (z, R)\) as the public key and private key, respectively.

\(\sigma \leftarrow \text{ARIS.Sig}(m, sk)\): Given a message \(m \in \{0, 1\}^*\) to be signed, this algorithm works as follows.

- Compute \((i'_1, \ldots, i'_k) \leftarrow \text{H}_1(m, z)\) where \(|i'_j| \leq |t|\) for \(j \in \{1, \ldots, k\}\).
- Compute \(r'_v \leftarrow \text{PRF}_2(z, i'_j)\) for \(j \in \{1, \ldots, k\}\), set \(r \leftarrow \sum_{i'=1}^k r'_i\).
- Retrieve \(R_{i'} \leftarrow R[i'_j]\) for \(j \in \{1, \ldots, k\}\), compute \(R \leftarrow \sum_{i'=1}^k R_{i'}\) and \(h \leftarrow \text{H}_2(R)\).
- Compute \((i_1, \ldots, i_k) \leftarrow \text{H}_3(m, h)\) (where \(|i_j| \leq |t|\)) and \(x_i \leftarrow \text{PRF}_1(z, i_j)\) for \(j \in \{1, \ldots, k\}\).
- Compute \(s \leftarrow r - \sum_{i=1}^k x_i\) and output \(\sigma \leftarrow (s, h)\).

\(\{0, 1\} \leftarrow \text{ARIS.Ver}(m, \sigma, pk)\): Given a message-signature pair \((m, \sigma)\) and \(pk\), this algorithm works as follows.

- Parse \((s, h) \leftarrow \sigma\) and compute \((i_1, \ldots, i_k) \leftarrow \text{H}_3(m, h)\), where \(|i_j| \leq |t|\) for \(j \in \{1, \ldots, k\}\).
- Retrieve \(Y_i \leftarrow Y[i_j]\) for \(j \in \{1, \ldots, k\}\) and set \(Y \leftarrow \sum_{i=1}^k Y_i\).
- Compute \(R' \leftarrow sP + Y\) and check if \(\text{H}_2(R') = h\) holds output \emph{valid}, and \emph{invalid} otherwise.
3.4 Security Analysis

We prove that ARIS is EU-CMA secure, as defined in Definition 2, in the Random Oracle Model (ROM) [39]. The proof uses the Forking Lemma [38].

Theorem 1. In the ROM, if adversary \( A \) can \((q_S, q_H)\)-break the EU-CMA security of ARIS after making \( q_H \) and \( q_S \) random oracles and signature queries, respectively; then we can build another algorithm \( B \) that runs \( A \) as a subroutine and can solve an instance of the ECDLP (as in Definition 3).

Proof. We let \( Y^* \leftarrow E(\mathbb{F}_p) \) be an instance the ECDLP for algorithm B to solve. On the input of \( Y^* \) and \( z \leftarrow \mathbb{Z}_q \), B works as follows.

Setup: B keeps three lists \( L_i \) for \( i \in \{1, 2, 3\} \) to keep track of the outputs of the random oracles and a list \( L_m \) to store the messages submitted to the sign oracle. B sets up the random oracle \( RO-Sim(\cdot) \) to handle the hash functions and generates the users’ public keys as follows.

- **Setup RO-Sim(\cdot)**: B implements \( RO-Sim(\cdot) \) to handle queries to hash functions \( H_1, H_2 \) and \( H_3 \), which are modeled as random oracles, as follows.

  - \( \alpha_1 \leftarrow RO-Sim(m, z, L_1) \): If \((m, z) \in L_1\), it returns the corresponding value \( \alpha_1 \). Else, it returns \( \alpha_1 \leftarrow \{0, 1\}^{l_1} \) as the answer and adds \((m, z, \alpha_1)\) to \( L_1 \).

  - \( \alpha_2 \leftarrow RO-Sim(R, L_2) \): If \( R \in L_2 \), it returns the corresponding value \( \alpha_2 \). Else, it returns \( \alpha_2 \leftarrow \{0, 1\}^{l_2} \) as the answer and adds \((R, \alpha_2)\) to \( L_2 \).

  - \( \alpha_3 \leftarrow RO-Sim(m, h, L_3) \): If \((m, h) \in L_3\), it returns the corresponding value \( \alpha_3 \). Else, it returns \( \alpha_3 \leftarrow \{0, 1\}^{l_3} \) as the answer and adds \((m, h, \alpha_3)\) to \( L_3 \).

- **Setup Public Key**: Given the parameters \((p, q, P, t, k)\), B works as follows to generate the user public key.
Select \( j \leftarrow [1,t] \) and sets the challenge public key element \( Y_j \leftarrow Y^* \).

Generate \( x_i \leftarrow Z_q \) for \( i \in \{1,\ldots,t\} \) and \( i \neq j \).

Compute \( Y_i \leftarrow x_iP \) for \( i \in \{1,\ldots,t\} \) and \( i \neq j \).

Set \( sk \leftarrow \{x_i\}_{i=1,i\neq j}^t \) and \( pk \leftarrow \{Y_1,\ldots,Y_t\} \).

\( \mathcal{A} \)'s Queries: \( \mathcal{A} \) queries the hash functions \( H_i \) for \( i \in \{1,2,3\} \) and the sign oracle for up to \( q_H \) and \( q_S \) times, respectively. \( B \) works as follows to handle these queries.

- **Hash Queries:** \( \mathcal{A} \)'s queries to hash functions \( H_1, H_2 \) and \( H_3 \) are handled by the \( RO-Sim(\cdot) \) function described above.

- **Signature Queries:** \( B \) works as follows to answer \( \mathcal{A} \)'s signature query on message \( m \). If \( m \in \mathcal{L}_m \), \( B \) retrieves the corresponding signature from \( \mathcal{L}_m \) and returns to \( \mathcal{A} \). Else, if \( m \notin \mathcal{L}_m \), it works as follows.

  1. Select \( s \leftarrow Z_q \) and compute \( S \leftarrow sP \).
  2. Select \( k \) indexes \( (i_1,\ldots,i_k) \leftarrow [1,\ldots,t] \).
  3. Set \( R \leftarrow S - \sum_{i=1}^k Y_i \) and \( \alpha_2 \leftarrow \{0,1\}^{l_2} \) and add \( (R,\alpha_2) \) to \( \mathcal{L}_2 \).
  4. If \( (\langle i_1,\ldots,i_k \rangle,h) \in \mathcal{L}_3 \) abort. Else, add \( (m,h,\langle i_1,\ldots,i_k \rangle) \) to \( \mathcal{L}_3 \).
  5. Output \( \sigma = (s,h) \) to \( \mathcal{A} \) and add \( (m,\sigma) \in \mathcal{L}_m \).

\( \mathcal{A} \)'s Forgery: Eventually, \( \mathcal{A} \) outputs a forgery \( \sigma^* = (s^*,h^*) \) on message \( m^* \) and public key \( pk \).

Following the EU-CMA definition (as in Definition 2), \( \mathcal{A} \) only wins the game if \( \text{ARIS.Ver}(m^*,\sigma^*,pk) \) returns \textit{valid} and \( m^* \) was never submitted to signature queries in the previous stage (i.e., \( m^* \notin \mathcal{L}_m \)).

\textit{Solving the ECDLP:} If \( \mathcal{A} \) does not output a valid forgery before making \( q_H \) hash queries and \( q_S \) signature queries, \( B \) also fails to solve the instance of ECDLP. Otherwise, if \( \mathcal{A} \) outputs a valid forgery
\((m^*, \sigma^* = (s^*, h^*))\), using the forking lemma, B rewinds \(A\) with the same random tape as in [38], to get a second forgery \((m', \sigma' = (s', h'))\) where, with an overwhelming probability \(s^* \neq s'\) and \(h^* = h'\). Based on [38, Lemma 1], \(H_3(m^*, h^*) \neq H_3(m^*, h')\), therefore, given \((m^*, h^*) \in \mathcal{L}_3\) and \((m^*, h') \in \mathcal{L}_3\), B can solve a random instance of the ECDLP problem (i.e., \(Y^*\)) if one of the following conditions hold.

- **Case 1:** For \((i_1^*, \ldots, i_k^*) \leftarrow H_3(m^*, h^*)\) and \((i_1', \ldots, i_k') \leftarrow H_3(m^*, h')\) we have \(j \in (i_1^*, \ldots, i_k^*)\) and \(j \notin (i_1', \ldots, i_k')\).

- **Case 2:** For \((i_1^*, \ldots, i_k^*) \leftarrow H_3(m^*, h^*)\) and \((i_1', \ldots, i_k') \leftarrow H_3(m^*, h')\) we have \(j \notin (i_1^*, \ldots, i_k^*)\) and \(j \in (i_1', \ldots, i_k')\).

If any of the above cases holds, B works as follows. If Case 1 holds, \(x_j \leftarrow s^* - \sum_{\eta=1, \eta \neq j}^k x_{i_n}^* - s' - \sum_{\eta=1}^k x_{i_n}' \mod p\). Else, if Case 2 hold, \(x_j \leftarrow s' - \sum_{\eta=1, \eta \neq j}^k x_{i_n}' - s^* - \sum_{\eta=1}^k x_{i_n}^* \mod p\). □

### 3.5 Performance Evaluation

We have fully implemented ARIS on FourQ curve [78] which is known to be the fastest EC that provides 128-bit of security. We provide implementations of ARIS on both commodity hardware and 8-bit microcontroller to evaluate its performance since most IoT applications are comprised of them both (e.g., commodity hardware as servers or control centers and microcontrollers as IoT devices connected to sensors). We compare the performance of ARIS with state-of-the-art digital signature schemes on both of these platforms, in terms of computation, storage and communication. Our implementation is open-sourced at the following link.

https://github.com/rbehnia/ARIS
3.5.1 Performance on Commodity Hardware

3.5.1.1 Hardware Configurations

We used a laptop equipped with Intel i7 Skylake processor @ 2.60 GHz and 12 GB RAM.

3.5.1.2 Software Libraries

We implemented ARIS using the open-sourced FourQ implementation [78], that offers the fastest EC operations, specifically EC additions that is critical for the performance of ARIS. We used an Intel processor as our commodity hardware and leveraged Intel intrinsics to optimize our implementation. Specifically, we implemented our PRF functions with Intel intrinsics (AES in counter mode). We used blake2 as our hash function [21] due to its efficiency.

We ran the open-source implementations of our counterparts on our hardware to compare their performance with ARIS.

3.5.1.3 Parameter Choice

Since we implement ARIS on FourQ curve, we use its parameters given in [78], which provide 128-bit security. Other than the curve parameters, the choice of $t, k$ also plays a crucial role for the security of ARIS. Specifically, $k$-out-of-$t$ combinations should also provide 128-bit security to offer this level of security overall. On the other hand, we can tune these parameters to achieve our desired security level with different performance trade-offs. If we increase $t$ and decrease $k$, this results in a larger storage with faster computations, and vice versa. For our commodity hardware implementation, we choose $t = 1024$ and $k = 18$, that we believe offers a reasonable trade-off between storage and computation as well as offering the desired 128-bit security level. We set $l_1 = 180$ and $l_2 = 256$. 
Table 3.1: Performance of ARIS and its counterparts on commodity hardware

<table>
<thead>
<tr>
<th>Scheme</th>
<th>Signing (µs)</th>
<th>Private Key† (KB)</th>
<th>Signature (KB)</th>
<th>Verification (µs)</th>
<th>Public Key (KB)</th>
<th>End-to-End Delay (µs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPHINCS [46]</td>
<td>13458</td>
<td>1.06</td>
<td>41000</td>
<td>370</td>
<td>1.03</td>
<td>13828</td>
</tr>
<tr>
<td>TACHYON [31]</td>
<td>138</td>
<td>0.016</td>
<td>4.4</td>
<td>18</td>
<td>864</td>
<td>156</td>
</tr>
<tr>
<td>RSA [181]</td>
<td>8083</td>
<td>0.75</td>
<td>0.41</td>
<td>48</td>
<td>0.38</td>
<td>8131</td>
</tr>
<tr>
<td>CEDA [161]</td>
<td>55</td>
<td>0.41</td>
<td>0.41</td>
<td>115</td>
<td>384.38</td>
<td>170</td>
</tr>
<tr>
<td>ECDSA [15]</td>
<td>725</td>
<td>0.03</td>
<td>0.06</td>
<td>927</td>
<td>0.03</td>
<td>1652</td>
</tr>
<tr>
<td>Ed25519 [45]</td>
<td>132</td>
<td>0.03</td>
<td>0.06</td>
<td>335</td>
<td>0.03</td>
<td>467</td>
</tr>
<tr>
<td>Kummer [44]</td>
<td>23</td>
<td>0.03</td>
<td>0.06</td>
<td>38</td>
<td>0.03</td>
<td>61</td>
</tr>
<tr>
<td>SchnorrQ [79]</td>
<td>12</td>
<td>0.03</td>
<td>0.06</td>
<td>22</td>
<td>0.03</td>
<td>34</td>
</tr>
<tr>
<td>ARIS</td>
<td>9</td>
<td>32.03</td>
<td>0.06</td>
<td>12</td>
<td>32</td>
<td>21</td>
</tr>
</tbody>
</table>

† System wide parameters (e.g., p,q,α) for each scheme are included in their corresponding codes, and private key size denote to specific private key size.

3.5.1.4 Experimental Results

We present the results of our experiments in Table 3.1. We observe that ARIS offers very fast signature generation and verification. It only takes 9 microseconds to generate a signature and 12 microseconds to verify it. This is the fastest among our counterparts, where the closest is SchnorrQ. Furthermore, if we use the same parameters set as for the AVR microcontroller, we can further speed up the signature generation to 6.5 microseconds, with the cost of a few microseconds on the verification speed. In SchnorrQ, a scalar multiplication is required in signature generation and a double scalar multiplication in verification. In ARIS, EC additions are required for signature generation and verification is done with a scalar multiplication and EC additions. This corresponds to a 33% faster signature generation and 83% faster verification for ARIS, compared to SchnorrQ. Therefore, we believe ARIS can be an ideal alternative for real-time applications.

ARIS signature size is the same with its EC-based counterparts [15, 44, 45, 79] , that is significantly lower than its RSA-based and hash-based counterparts [46, 161, 181]. On the other hand, ARIS comes with a larger private and public key, that is 32 KB.
Table 3.2: Performance of ARIS and its counterparts on 8-bit AVR

<table>
<thead>
<tr>
<th>Scheme</th>
<th>Signing</th>
<th>Private Key</th>
<th>Signature</th>
<th>Verification</th>
<th>Public Key</th>
<th>End-to-End Delay</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(s)</td>
<td>(KB)</td>
<td>(KB)</td>
<td>(s)</td>
<td>(KB)</td>
<td>(s)</td>
</tr>
<tr>
<td>ECDSA [15]</td>
<td>1.77</td>
<td>0.03</td>
<td>0.06</td>
<td>1.80</td>
<td>0.03</td>
<td>3.57</td>
</tr>
<tr>
<td>Ed25519 [45, 125]</td>
<td>1.45</td>
<td>0.03</td>
<td>0.06</td>
<td>2.06</td>
<td>0.03</td>
<td>3.51</td>
</tr>
<tr>
<td>μKummer [44, 178]</td>
<td>0.65</td>
<td>0.03</td>
<td>0.06</td>
<td>1.02</td>
<td>0.03</td>
<td>1.67</td>
</tr>
<tr>
<td>SchnorrQ [79, 142]</td>
<td>0.27</td>
<td>0.03</td>
<td>0.06</td>
<td>0.60</td>
<td>0.03</td>
<td>0.87</td>
</tr>
<tr>
<td>ARIS</td>
<td>0.19</td>
<td>16</td>
<td>0.06</td>
<td>0.37</td>
<td>8</td>
<td>0.56</td>
</tr>
</tbody>
</table>

3.5.2 Performance on 8-bit AVR

3.5.2.1 Hardware Configurations

We used an 8-bit AVR ATmega 2560 microcontroller as our IoT device to implement ARIS. ATmega 2560 is equipped with 256 KB flash memory, 8 KB SRAM and 4 KB EEPROM, with a maximum clock frequency of 16 MHz. ATmega 2560 is extensively used in practice for IoT applications (especially in medical implantables) due to its energy efficiency [194].

3.5.2.2 Software Libraries

We implemented ARIS on ATmega 2560 using the 8-bit AVR implementation of FourQ curve [142], that provides the basic EC operations and a blake2 hash function. We implemented our scheme with IAR embedded workbench and used its cycle-accurate simulator for our benchmarks.

As for our counterparts, we used their open-sourced implementations [125, 142, 151, 178]. Note that we only compare ARIS with its EC-based counterparts, due to their communication and storage efficiency. Moreover, resource-constrained processors such as ATmega 2560 may not be suitable for heavy computations (e.g., exponentiation with 3072-bit numbers in RSA [181] and CEDA [161]).
3.5.2.3 Parameter Choice

As mentioned, ARIS can be instantiated with different $t, k$ values that offers a trade-off between storage and computation. Since ATmega 2560 is a storage-limited device, we select our parameters as $t = 256$ and $k = 28$ to offer storage efficiency. Moreover, this allows us to store the private components ($x_i$ and $r_i$), instead of deterministically generating them at signature generation, and still have a tolerable storage even for an 8-bit microcontroller. We also set $l_1 \leftarrow 224$ and $l_2 \leftarrow 256$.

3.5.2.4 Experimental Results

Table 3.2 shows the performance of ARIS compared with its counterparts. The speed improvements of ARIS can also be observed for ATmega 2560. ARIS is 42% faster in signature generation and 76% faster in signature verification compared to its closest counterpart [79]. This can translate into a significant practical difference when considered real-time applications that require fast authentication. Note that these benchmarks are obtained with a more “storage friendly” parameter choice, and can be further accelerated with different parameter choices where the microcontroller is not memory-constraint.

One may notice that due to our parameter choice, the key sizes in our 8-bit microcontroller implementation are smaller. As aforementioned, this is because we select a different parameter set for $t, k$. Moreover, we store the private components as well, that correspond the 8 KB of the signer storage. Since we store these keys on the flash memory of ATmega 2560, they only correspond to 6% and 3% of the total memory, for private key and public key, respectively. Therefore, although we have significantly larger keys than our EC-based counterparts, it is still feasible to store them even on highly resource-constrained 8-bit microcontrollers.
3.5.2.5 Energy Efficiency

It is highly desirable to minimize the energy consumption of cryptographic primitives in IoT applications to offer a longer battery life. For microcontrollers, energy consumption of the device can be measured with the formula \( E = V \times I \times t \), where \( V \) is voltage, \( I \) is current and \( t \) is the computation time [19]. Considering that the voltage and the current of a microcontroller are constant when the device is active, the energy consumption linearly increases with the computation time. Since ARIS offers the fastest signature generation and verification, energy consumption of ARIS is the lowest among its counterparts, and therefore would be preferred in applications that require longer battery life.

3.6 Conclusion

In this chapter, we presented a new efficient signature scheme to meet the strict minimum delay requirements of some real-time IoT systems. This is achieved by harnessing the homomorphic property of the underlying ECDLP-based one-way function and the precomputation technique proposed in [63]. Our experimental results showed that the proposed scheme outperforms its state-of-the-art counterparts in signing and verification speed as well as in energy efficiency. The proposed scheme is shown to be secure, in the Random Oracle Model, under the hardness of the ECDLP. We open-sourced our implementation to enable public testing and verification.
Chapter 4: Compatible Certificateless and Identity-Based Cryptosystems for Heterogeneous IoT

4.1 Introduction

Mobile and heterogeneous IoT applications harbor large quantities of resource-limited and non-stationary IoT devices, each with different capabilities, configurations, and user domains. For instance, emerging commercial aerial drone network protocols\(^4\) need a near real-time communication and processing over a bandwidth-limited network. There are multiple hurdles of relying on traditional PKI for such systems: (i) The maintenance of PKI for such IoT networks demands a substantial infrastructure investment [186]. (ii) PKI requires transmission and verification of certificate chains at the sender’s/verifier’s side. This communication and computation overhead could create a major bottleneck for mobile IoT devices (e.g., aerial drones [162]) that potentially need to interact with a number of devices. In certain cases, these certificate chains might be larger than the actual measurements/commands being transmitted and therefore, might be the dominating cost for these applications. Figure 4.1-a depicts a high-level illustration of traditional PKI for mobile IoT applications.

Identity-based (IDB) and certificateless (CL) cryptosystems offer implicit certification [8, 53, 186], and therefore can mitigate the aforementioned hurdles. In IDB, the user’s public key is derived from their identifying information, and the system relies on a fully-trusted third party (TTP), called the private key generator (PKG), to issue users’ private keys. The top portion of Figure 4.1-b

\(^3\)This chapter was published in [33]. Permission is included in Appendix A.

\(^4\)https://github.com/mavlink/mavlink/mavlink
Figure 4.1: Proposed IDB and CL cryptosystems and alternatives (high-level)

depicts IDB encryption, wherein the user authenticates itself to the PKG and receives a private key corresponding to its identity $D_1$. The sender can use $D_1$ as the public key to run encryption. IDB is potentially suitable for applications where the system setup is done and managed by a trusted centralized entity. In CL systems [8] the trust on the TTP is lowered by allowing the private key of the user to consist of two parts. One is computed by the user and the other is by the TTP (called the KGC). The bottom portion of Figure 4.1-b outlines CL encryption, where the user computes its key pair and then works as in IDB to receive the other part of the private key from the KGC. CL cryptosystems are suitable for architectures that might not assume a fully trusted third party where the trust level on the KGC is similar to traditional certification authorities.

IDB and CL cryptosystems have their own merits and drawbacks, and therefore might be used in different IoT applications. Hence, it is expected that there will be different user groups who
rely on IDB and CL cryptosystems initiated in different domains/systems. For example, Amazon’s Prime Air\(^5\) would require drones, under the complete control of Amazon, to interact with other drones (e.g., personal) to ensure safe operation. By employing IDB cryptography on its drones, Amazon can have complete control over the operations of its delivery drones while avoiding the overheads of traditional PKI. However, it is a strong assumption that other drones, outside Amazon’s network, will adopt a similar cryptographic setting to ensure safe and secure operations. For instance, personal users rarely trust any third party to have complete control and knowledge of their drones’ activity. To the best of our knowledge, there is a significant research gap in enabling a seamless communication between users who are registered under different domains (e.g., IDB and CL). This is a potential obstacle to widely deploy efficient certificate-free solutions in heterogeneous environments. This limitation is mentioned in Figure 4.1-b. Moreover, it is important to further improve the computational efficiency of IDB and CL techniques to offer a low end-to-end delay that is needed by delay-aware IoT applications.

4.1.1 Our Contribution

We propose a new series of public key encryption, digital signature, and key exchange schemes that permit users from different domains (IDB or CL) to communicate seamlessly. To our knowledge, this is the first set of certificate-free cryptosystems that achieve such compatibility and efficiency, and therefore a suitable alternative for resource-limited IoT systems such as commercial aerial drones. The idea behind our constructions is to create special key generation algorithms that harness the additive homomorphic property of the exponents and cover-free functions to enable the users to incorporate their private keys into the one provided by the TTP without falsifying it. As detailed in §4.3, this special design is applicable across our IDB and CL algorithms, and therefore it permits a

\(^5\)https://www.amazon.com/Amazon-Prime-Air/b?ie=UTF8&node=8037720011
seamless communication between our IDB and CL cryptosystems. This strategy also reduces the cost of online operations and enables our schemes to achieve a lower end-to-end delay compared to their counterparts. We elaborate on some desirable properties of our schemes as below.

- **Compatible IDB and CL Schemes:** Figure 4.1.c outlines the concept of compatible IDB and CL schemes where the users from different domains (and trust-levels) can use identical encryption, signature, and key exchange algorithms to communicate without any additional overhead.

- **Computation and Communication Efficiency:** Based on our analysis, new schemes offer performance advantages over their counterparts: (i) Similar to other IDB/CL cryptosystems, our schemes lift the hurdle of certificate transmission and verification, and therefore offer significant communication efficiency over some of the most efficient PKI-based schemes. This advantage grows proportional to the size of the certificate chain. (ii) Our schemes outperform their certificate-free counterparts on the vast majority of the performance metrics. For instance, the end-to-end delay in our IDB/CL encryption schemes is $\approx 25\%$ lower than our most efficient counterpart in [208]. Our signature schemes achieve up to $52\%$ faster end-to-end delay as compared to our counterparts. We also achieve a $65\%$ lower end-to-end delay for our key exchange schemes.

- **Open-Sourced Implementation:** We implemented our schemes on a commodity hardware and an 8-bit AVR microprocessor, and compared their performance with a variety of their counterparts capturing some of the most efficient traditional PKI, IDB and CL schemes (see §4.5 for details). We open-source our implementations for broad testing, benchmarking, and adoption purposes.
4.2 System and Security Model

In this section, we present the definitions of identity-based and certificateless encryption and signature schemes followed by the security models for certificateless encryption and signature schemes.

4.2.1 System Models

**Definition 13.** An identity-based encryption scheme is consisted of four algorithms \( IBE = \{\text{Setup}, \text{Extract}, \text{Enc}, \text{Dec}\} \).

\((msk, params) \leftarrow IBE.\text{Setup}(1^\kappa):\) Given the security parameter \( \kappa \), the PKG selects master secret key \( msk \), computes master public key \( mpk \) and system parameters \( params \) (an implicit input to all the following algorithms).

\((sk_{ID}, Q_{ID}) \leftarrow IBE.\text{Extract}(ID, msk):\) Given an identity \( ID \) and \( msk \), the PKG computes the commitment value \( Q_{ID} \) and the private key \( sk_{ID} \).

\( c \leftarrow IBE.\text{Enc}(m, ID, Q_{ID}):\) Given a message \( m \) and \((ID, Q_{ID})\), the sender computes the ciphertext \( c \).

\( m \leftarrow IBE.\text{Dec}(sk_{ID}, c):\) Given the ciphertext \( c \) and the private key of the receiver \( sk_{ID} \), the receiver returns either the corresponding plaintext \( m \) or \( \perp \) (invalid).

**Definition 14.** An identity-based signature scheme is defined by four algorithms \( IBS = \{\text{Setup}, \text{Extract}, \text{Sig}, \text{Ver}\} \).

\((msk, mpk, params) \leftarrow IBS.\text{Setup}(1^\kappa):\) As in \( IBE.\text{Setup} \) in Definition 13.

\((sk_{ID}, Q_{ID}) \leftarrow IBS.\text{Extract}(ID, msk):\) As in \( IBE.\text{Extract} \) in Definition 13.

\( \sigma \leftarrow IBS.\text{Sign}(m, sk_{ID}):\) Given a message \( m \) and \( sk_{ID} \), returns a signature \( \sigma \).
\[ d \leftarrow \text{IBS.Verify}(m, ID, Q_{ID}, \sigma) \]: Given \( m, \sigma \) and \((ID, Q_{ID})\) as input, it outputs a decision bit \( d \in \{0, 1\} \).

Definition 15. A certificateless encryption scheme is defined by six algorithms \( \text{CLE} = \{ \text{KGCSetup}, \text{UserSetup}, \text{PartKeyGen}, \text{UserKeyGen}, \text{Enc}, \text{Dec} \} \).

\((\text{msk}, \text{mpk}, \text{params}) \leftarrow \text{CLE.KGCSetup}(1^\kappa)\): Give the security parameter \( \kappa \), the KGC generates master secret key \( \text{msk} \), master public key \( \text{mpk} \) and the system parameters \( \text{params} \) (an implicit input to all the following algorithms).

\((\alpha, U) \leftarrow \text{CLE.UserSetup}(\cdot)\): The user \( ID \) computes her secret value \( \alpha \) and its corresponding commitment \( U \).

\((w, Q_{ID}) \leftarrow \text{CLE.PartKeyGen}(ID, U, \text{msk})\): Given \( ID, U, \) and \( \text{msk} \), the KGC computes partial private key \( w \) and its corresponding public commitment \( Q_{ID} \).

\(x_{ID} \leftarrow \text{CLE.UserKeyGen}(w, \alpha)\): Given \((w, \alpha)\), the user \( ID \) computes \( x_{ID} \).

\(c \leftarrow \text{CLE.Enc}(m, ID, Q_{ID})\): Given \((m, ID, Q_{ID})\), sender computes ciphertext \( c \).

\(m' \leftarrow \text{CLE.Dec}(x_{ID}, c)\): Given the ciphertext \( c \) and the private key of the receiver \( x_{ID} \), the receiver returns either corresponding plaintext \( m \) or \( \bot \) (invalid).

Definition 16. A certificateless signature scheme is defined by six algorithms \( \text{CLS} = \{ \text{KGCSetup}, \text{UserSetup}, \text{PartKeyGen}, \text{UserKeyGen}, \text{Sig}, \text{Ver} \} \). The definition of algorithms are as in Definition 15 except for \((\text{CLS.Sig}, \text{CLS.Ver})\).

\(\sigma \leftarrow \text{CLS.Sig}(m, x_{ID})\): Given a message \( m \), and the signer’s private key \( x_{ID} \), it returns a signature \( \sigma \).

\[ d \leftarrow \text{CLS.Ver}(m, ID, Q_{ID}, \sigma) \]: Given \( m, \sigma \) and \((ID, Q_{ID})\) as input, it outputs a decision bit \( d \in \{0, 1\} \).
4.2.2 Security Models

The security model of identity-based schemes is slightly stronger than those for traditional PKI based schemes. More specifically, the adversary can query for the private key of any user $ID$, except for the target user $ID^\ast$. In this chapter, we constructed our schemes by following the security model of Identity-based systems proposed in [53]. In certificateless systems, the private key of the users consists of two parts: (i) user secret key $\alpha$, which is selected by the user, and (ii) partial private key $w$, which is supplied to the user by the KGC. Therefore, following [8], it is natural to consider two types of adversaries for such systems.

A Type-I adversary $A_I$ does not have access to $msk$ or the user’s partial private key $w$ but is able to replace any user’s public key $U$ with public key of its choice $U'$. However, in our security model, since we adopt the binding method [8], replacing the public key will result in falsifying the partial private key (and evidently the private key). Therefore, following [20], we allow $A_I$ to query for the secret key of the user via $\alpha \leftarrow \mathcal{O}_{\text{SecKey}}(ID)$. Note that our model can also be extended to allow $A_I$ to replace the public key of the user (see §4.4). A Type-II adversary $A_{II}$ is assumed to be a malicious KGC. Having knowledge on $msk$, $A_{II}$ can query the partial private key of the user via $w \leftarrow \mathcal{O}_{\text{PartKey}}(ID)$. Following [8], we allow the adversary $A \in \{A_I, A_{II}\}$ to extract private key of users’ private keys via the $x_{ID} \leftarrow \mathcal{O}_{\text{Corrupt}}(ID)$. We note that inspired by [22], many improvements on the security models of certificateless systems have been suggested (e.g., [20, 22, 122]). In this chapter, we provide our proof in the original model proposed in [8, 22], but note that many of those stronger security requirements can be enforced if needed.

Definition 17. The indistinguishability of a CLE under chosen ciphertext attack (IND-CLE-CCA) experiment $\text{Expt}_{\mathcal{A}}^{\text{IND-CLE-CCA}}$ is defined as follows.

- $\mathcal{C}$ runs $\text{CLE.KGCSsetup}(1^\kappa)$ and returns $mpk$ and $params$ to $\mathcal{A}$.  

34
\( (ID^*, m_0, m_1) \leftarrow A^{O_{PartKey}, O_{SecKey}, O_{Corrupt}, O_{Dec}}(mpk, params) \)

- \( C \) picks \( b \leftarrow \{0, 1\} \), \( c_b \leftarrow \text{CLE.Enc}(m_b, ID^*, params) \) and returns \( c_b \) to \( A \).

- \( A \) performs the second series of queries, with a restriction of querying \( ID^* \) or \( c_b \) to \( O_{Corrupt}(\cdot) \) or \( \text{CLE.Dec}(\cdot) \), respectively. Finally, \( A \) outputs a bit \( b' \).

\( A \) wins the above experiment if \( b = b' \) and the following conditions hold: (i) \( ID^* \) was never submitted to \( O_{Corrupt} \). (ii) If \( A = A_I \), \( ID^* \) was never submitted to \( O_{PartKey} \). (iii) If \( A = A_{II} \), \( ID^* \) was never submitted to \( O_{SecKey} \). The IND-CLE-CCA advantage of \( A \) is \( \Pr[b = b'] \leq \frac{1}{2} + \epsilon \), for a negligible \( \epsilon \).

**Definition 18.** The existential unforgeability under chosen message attack (EU-CLS-CMA) experiment \( \text{Expt}_A^{EU-CLS-CMA} \) for a certificateless signature CLS is defined as follows.

- \( C \) runs \( \text{CLS.KGCSetup}(1^*) \) and returns \( mpk \) and \( params \) to \( A \).

- \( (ID^*, m^*, \sigma^*) \leftarrow A^{O_{PartKey}, O_{SecKey}, O_{Corrupt}, O_{Sign}}(mpk, params) \)

\( A \) wins the above experiment if \( 1 \leftarrow \text{CLS.Ver}(m^*, \sigma^*, ID) \), and the following conditions hold: (i) \( ID^* \) was never submitted to \( O_{Corrupt} \). (ii) If \( A = A_I \), \( ID^* \) was never submitted to \( O_{PartKey} \). (iii) If \( A = A_{II} \), \( ID^* \) was never submitted to \( O_{SecKey} \). The EU-CLS-CMA advantage of \( A \) is \( \Pr[\text{Expt}_A^{EU-CLS-CMA} = 1] \).

### 4.3 Proposed Schemes

#### 4.3.1 Proposed Identity-Based Cryptosystem

Most of pairing-free IDB schemes rely on the classical signatures (e.g., [182]) in their key generation to provide implicit certification. The use of such signatures to construct IDB schemes
usually require several expensive operations (e.g., scalar multiplication), and therefore may incur a non-negligible computation overhead. To reduce this cost, we exploit the message encoding technique and subset resilient functions (similar to [179]) along with the exponent product of powers property to generate keys. This permits an improved efficiency for both the PKG and user since it only requires a hash call and a few point additions. Before we present our schemes, we define five hash functions $H_1: E(\mathbb{F}_p) \times E(\mathbb{F}_p) \rightarrow \{0, 1\}$, $H_2: \{0, 1\}^n \times \{0, 1\}^* \rightarrow \mathbb{Z}_q$, $H_3: E(\mathbb{F}_p) \rightarrow \{0, 1\}^n$, $H_4: \{0, 1\}^n \rightarrow \{0, 1\}^n$, and $H_5: \{0, 1\}^n \times E(\mathbb{F}_p) \rightarrow \mathbb{Z}_q$, where all hash functions are random oracles [34].

Our IDB schemes use similar IBE.Setup and IBE.Extract functions whose key steps are outlined as follows. In the IBE.Setup, the PKG selects $t$ values $v_i \leftarrow \mathbb{Z}_q$, and computes their commitments as $V_i \leftarrow v_i P \mod p$, for $i = 1, \ldots, t$, it then sets the master secret key $msk \leftarrow (v_1, \ldots, v_t)$ and the system-wide public key $mpk \leftarrow (V_1, \ldots, V_t)$. This is similar to the scheme in [179], where EC scalar multiplication is used as the one-way function. In IBE.Extract, the PKG picks a nonce $\beta \leftarrow \mathbb{Z}_q$ and computes its commitments $Q \leftarrow \beta P \mod p$. The PKG then derives indexes $(j_1, \ldots, j_k) \leftarrow H_1(ID, Q)$, which select $k$-out-of-$t$ elements from the master secret key $v_{j_i}$ for $i = 1, \ldots, k$. Note that $Q$ is implicitly authenticated by being included in input of $H_1(\cdot)$, this is similar to the technique used in other pairing-free identity-based and certificateless systems [22, 99]. In Steps 3-4, unlike the scheme in [179], where secret keys are exposed, we use the additive homomorphic property in the exponent to mask the one-time signature $y$ (Step 3) via the nonce $\beta$ (in line with [27, 31]). The PKG will then sends $(x, Q)$ to the user via a secure channel.

4.3.1.1 Identity-Based Encryption Scheme

In IBE.Enc (Algorithm 2, Step 2), the indexes obtained from $H_1$ are used to retrieve the components $V_{j_i}$ from the system-wide public key $mpk$. The input of $H_3$ is the ephemeral key, which
Algorithm 2 Identity-Based Encryption

\[(msk, params) \leftarrow IBE.\text{Setup}(1^\kappa)\]: Bob encrypts message \(m \in \{0,1\}^n\).

1. \(\sigma \leftarrow \{0,1\}^n, r \leftarrow H_2(\sigma, m), R \leftarrow rP \mod p\)
2. \((j_1, \ldots, j_k) \leftarrow H_1(ID_a, Q_a), Y_a \leftarrow \sum_{i=1}^{k} V_{j_i} \mod p\)
3. \(u \leftarrow H_3(r(Y_a + Q_a) \mod p) \oplus \sigma, v \leftarrow H_4(\sigma) \oplus m\)
4. return \(c = (R, u, v)\)

\((w, Q) \leftarrow IBE.\text{Extract}(ID, U, msk)\):

1. \(\beta \leftarrow \{0,1\}^n, Q \leftarrow \beta P \mod p\)
2. \((j_1, \ldots, j_k) \leftarrow H_1(ID, Q)\) where for all \(i = 1, \ldots, t, 1 < j_i < |t|\)
3. \(y \leftarrow \sum_{i=1}^{k} v_{j_i} \mod q\)
4. \(x \leftarrow y + \beta \mod q\)
5. return \((x, Q)\)

\((msk, params) \leftarrow IBE.\text{Setup}(1^\kappa)\):

1. Select primes \(p\) and \(q\) and \((t, k) \in \mathbb{N}\) where \(t >> k\).
2. for \(i = 1, \ldots, t\) do
3. \(v_i \leftarrow \mathbb{Z}_q, V_i \leftarrow v_i P \mod p\)
4. return \(msk \leftarrow (v_1, \ldots, v_t), mpk \leftarrow (V_1, \ldots, V_t)\) and \(params \leftarrow (H_1, H_2, H_3, H_4, p, q, k, t, mpk)\)

4.3.1.2 Identity-Based Signature Scheme

In \(IBS.\text{Verify}\), the public key of the user \(Y_a\) is computed from \(V_{j_i} \in mpk\) via the indexes retrieved from the output of \(H_1\). The key generation is as in Algorithm 2. The rest of the signing and verification steps are akin to Schnorr signatures [182].

4.3.1.3 Identity-Based Key Exchange Scheme

For the key exchange scheme, we run \(IBE.\text{Setup}\) and then let both parties, Alice and Bob, obtain \((x_A, Q_A)\) and \((x_B, Q_B)\) via the \(IBE.\text{Extract}\) algorithm, respectively. Alice then picks \(z_A \leftarrow \mathbb{Z}_q\), computes its commitment \(M_A \leftarrow z_A P \mod p\), and sends \((M_A, Q_A)\) to Bob. Bob does the same and sends \((M_B, Q_B)\) to Alice. Alice then computes \((j_1, \ldots, j_k) \leftarrow H_1(ID_b, Q_b)\) and \(Y_b \leftarrow \sum_{i=1}^{k} V_{j_i} \mod p\).
Algorithm 3 Identity-Based Signature

\[(msk, params) \leftarrow IBS.\text{Setup}(1^\kappa):\] Description identical to IBE.\text{Setup} in Algorithm 2, except that only the description of \(H_1\) and \(H_5\) is included in \(params\).

\[(w,Q) \leftarrow IBS.\text{Extract}(ID,U,msk):\] As in IBE.\text{Extract} in Algorithm 2.

\[(s,e) \leftarrow IBS.\text{Sign}(m,x_a):\] Alice IDa signs message \(m\).

\[
\begin{align*}
1: & \quad r \leftarrow Z_q, R \leftarrow rP \mod p \\
2: & \quad e \leftarrow H_5(m,R) \\
3: & \quad s \leftarrow r - e \cdot x_a \mod q \\
4: & \quad \text{return} (s,e)
\end{align*}
\]

\[
\{0,1\} \leftarrow IBS.\text{Verify}(m,ID_a,Q_a,\langle s,e \rangle):\] Bob verifies the signature \((s,e)\).

\[
\begin{align*}
1: & \quad (j_1, \ldots, j_k) \leftarrow H_1(ID_a,Q_a) \\
2: & \quad Y_a \leftarrow \sum_{i=1}^k V_{j_i} \mod p \\
3: & \quad R' \leftarrow sP + e(Y_a + Q_a) \mod p \\
4: & \quad \text{if } e = H_5(m,R') \text{ then return 1} \\
5: & \quad \text{else return 0}
\end{align*}
\]

and outputs the shared secret key as \(K_a \leftarrow x_a(Y_b + Q_b) + z_aM_b \mod p\). Bob works similarly, and outputs the shared key as \(K_b \leftarrow x_b(Y_a + Q_a) + z_bM_a \mod p\).

4.3.2 Proposed Certificateless Cryptosystem

For our CL schemes to achieve the same trust level (Level 3) [107] on the third party (KGC), as in traditional PKI, we use the binding method [8] in the CLE.\text{PartKeyGen} and CLS.\text{PartKeyGen} algorithms. Note that the same secure channel which is used for user authentication (e.g., SSL/TLS), can be used to send the user commitment \(U\) to the KGC. This permits an implicit certification of \(U\), and therefore any changes of \(U\), will falsify the private key.

The CLE.\text{KGCSetup} algorithm is as in IBE.\text{Setup} in Algorithm 2. The CLE.\text{PartKeyGen} algorithm is similar to the IBE.\text{Extract} in Algorithm 2, with the difference that the user commitment \(U\) is used to compute \(Q\). In CLE.\text{UserKeyGen}, the correctness of the partial private key is checked first before the private key \(x\) is computed.
Algorithm 4 Certificateless Encryption

\[(msk, params) \leftarrow CLE.KGCSetup(1^n)\]:
As in IBE.Setup in Alg. 2.

\[(\alpha, U) \leftarrow CLE.UserSetup(\cdot)\):
1: \(\alpha \leftarrow \mathbb{Z}_q, U \leftarrow \alpha P \mod p\)
2: return \((\alpha, U)\)

\((w, Q) \leftarrow CLE.PartKeyGen(ID, U, msk)\):
1: \(\beta \leftarrow \mathbb{Z}_q, W \leftarrow \beta P \mod p\)
2: \(Q = U + W \mod p\)
3: \((j_1, \ldots, j_k) \leftarrow H_1(ID, Q)\) where for all \(i = 1, \ldots, t, 1 < j_i < |t|\)
4: \(y \leftarrow \sum_{i=1}^{k} v_{j_i} \mod q\)
5: \(w \leftarrow y + \beta \mod q\)
6: return \((w, Q)\)

\(x \leftarrow CLE.UserKeyGen(w, \alpha)\):
1: \((j_1, \ldots, j_k) \leftarrow H_1(ID, Q), Y \leftarrow \sum_{i=1}^{k} V_{j_i} \mod p\)
2: \(W' \leftarrow Q - U \mod p, W'' := wP - Y \mod p\)
3: if \(W' = W''\) then return \(x \leftarrow w + \alpha \mod q\) else return \(\perp\)

\(c \leftarrow CLE.Enc(m, ID_a, Q_a)\): Bob encrypts message \(m \in \{0,1\}^n\).
1: \(\sigma \leftarrow \mathbb{Z}_q, r \leftarrow H_2(\sigma, m), R \leftarrow rP \mod p\)
2: \((j_1, \ldots, j_k) \leftarrow \mathbb{H}_1(ID_a, Q_a), Y_a \leftarrow \sum_{i=1}^{k} V_{j_i} \mod p\)
3: \(u \leftarrow H_3(r(Y_a + Q_a) \mod p) \oplus \sigma, v \leftarrow H_4(\sigma) \oplus m\)
4: return \(c = (R, u, v)\)

\(m \leftarrow CLE.Dec(x_a, c)\): Alice decrypts the ciphertext \(c\).
1: \(\sigma' \leftarrow H_3(x_a R \mod p) \oplus u\)
2: \(m' \leftarrow v \oplus H_4(\sigma'), r' \leftarrow H_2(\sigma', m)\)
3: if \(r' P \mod p = R\) then return \(m'\) else return \(\perp\)

4.3.2.1 Certificateless Encryption Scheme

Note that the CLE.Enc and CLE.Dec algorithms are identical to IBE.Enc and IBE.Dec algorithms in Algorithm 2.

4.3.2.2 Certificateless Signature Scheme

The setup and key generation algorithms are as in Algorithm 4, and the CLS.Sign and CLS.Verify algorithms are as in IBS.Sign and IBS.Verify in Algorithm 3, respectively.

4.3.2.3 Certificateless Key Exchange Scheme

Given the compatibility of our IDB and CL schemes, after the initial algorithms (system setup and key generation) take place as in Algorithm 4, the CL key exchange will be identical to the one proposed in the identity-based key exchange scheme above.
Algorithm 5 Certificateless Digital Signature

\[
\begin{align*}
(msk, params) \leftarrow \text{CLS.KGCSetup}(1^n) : & \quad \text{As in CLE.KGCSetup in Alg. 4, except that } H_1 \text{ and } H_5 \text{ are in } \text{params.} \\
(\alpha, U) \leftarrow \text{CLS.UserSetup}(params) : & \quad \text{As in CLE.UserSetup in Alg. 4.} \\
(w, Q) \leftarrow \text{CLS.PartKeyGen}(ID, U, msk) : & \quad \text{As in CLE.PartKeyGen in Alg. 4.} \\
x \leftarrow \text{CLS.UserKeyGen}(params, \alpha, w) : & \quad \text{As in CLE.UserKeyGen in Alg. 4.}
\end{align*}
\]

\[
\begin{align*}
(s, e) \leftarrow \text{CLS.Sign}(m, x_a) : & \quad \text{Alice } ID_a \text{ signs message } m. \\
1: & \quad r \overset{s}{\leftarrow} \mathbb{Z}_q, R \leftarrow rP \mod p \\
2: & \quad e \leftarrow H_5(m, R) \\
3: & \quad s \leftarrow r - e \cdot x_a \mod q \\
4: & \quad \text{return } (s, e)
\end{align*}
\]

\[
\begin{align*}
\{0, 1\} \leftarrow \text{CLS.Verify}(m, Q_a, (s, e)) : & \quad \text{Bob verifies the signature } (s, e). \\
1: & \quad (j_1, \ldots, j_k) \leftarrow H_1(ID_a, Q_a) \\
2: & \quad Y_a \leftarrow \sum_{i=1}^k V_{j_i} \mod p \\
3: & \quad R' \leftarrow sP + e(Y_a + Q_a) \mod p \\
4: & \quad \text{if } e = H_5(m, R') \text{ then return } 1 \\
5: & \quad \text{else return } 0
\end{align*}
\]

4.3.3 Compatibility of Identity-Based and Certificateless Schemes

In our CL schemes, we utilize the additive homomorphic property of the exponents (i.e., \(w\)) when the KGC includes the addition of commitments (\(W\) and \(U\)) in the \(H_1\). After receiving \(w\), the user exploits the homomorphic property to modify the key without falsifying it and obtain \(x\). For instance, we observed that our counterparts (e.g., [8, 53]) do not offer such a compatibility, since the partial private key is the KGC’s commitment to the (hash of) user identity, without a homomorphic property. Moreover, the KGC does not output any auxiliary value to incorporate the user commitment with it.

As shown above, our IDB and CL schemes are compatible, thanks to the special design of their key generation algorithms (i.e., \texttt{Extract} in IDB, \texttt{UserSetup} and \texttt{PartKeyGen} in CL). Therefore, after the users computed/obtained their keys from the third party, the interface of the main cryptographic functions (e.g., encrypt, decrypt, sign, etc.) are identical in both systems, therefore, the users can communicate with uses in different domains seamlessly. For instance, ciphertext \(c = (R, u, v)\) outputted by the \texttt{CLE.Enc} in Algorithm 2, can be decrypted by a user in the identity-based setting.
4.4 Security Analysis

Theorem 2. If an adversary $\mathcal{A}_I$ can break the IND-CLE-CCA security of the encryption scheme proposed in Algorithm 4 after $q_{H_i}$ queries to random oracles $H_i$ for $i \in \{1,2,3,4\}$, $q_D$ queries to the decryption oracle and $q_{sk}$ to the private key extraction oracle with probability $\epsilon$. There exists another algorithm $\mathcal{C}$ that runs $\mathcal{A}_I$ as subroutine and breaks a random instance of the CDH problem $(P,aP,bP)$ with probability $\epsilon'$ where:

$$\epsilon' > \frac{1}{q_{H_3}} \left( \frac{2\epsilon}{q_{H_3}^2} - \frac{q_{H_3}(q_{H_3}^2 + 1)}{2^{n_q}} - \frac{2q_D}{p} \right).$$

Proof. Our proof technique is similar to the one in [22]. $\mathcal{C}$ simulates the real environment for $\mathcal{A}_I$. It knows the $t$ secret values $v_i'$s in the scheme, and tries to embed a random instance of the CDH problem $(P,aP,bP)$. $\mathcal{C}$ sets $aP$ as a part of the target user's ($ID^*$) public key (i.e., $Q_{ID^*} \leftarrow aP$) and $bP$ as a part of the challenge ciphertext (i.e., $R^* \leftarrow bP$). $\mathcal{C}$ uses four lists, namely $\text{List}_{H_1}$, $\text{List}_{H_2}$, $\text{List}_{H_3}$, and $\text{List}_{H_4}$, to keep track of the random oracle responses and following the IND-CLE-CCA experiment $\text{Expt}_{\mathcal{A}}^{\text{IND-CLE-CCA}}$ (Definition 17), $\mathcal{C}$ responds to $\mathcal{A}_I$ queries as follows.

Queries to $H_1(ID_i, Q_i)$: If the entry $(ID_i, Q_i, h_{1,i})$ exists in $\text{List}_{H_1}$, $\mathcal{C}$ returns $h_{1,i}$, otherwise, it chooses $h_{1,i} \leftarrow \gamma$, and inserts $(ID_i, Q_i, h_{1,i})$ in $\text{List}_{H_1}$.

Queries to $H_2(\sigma_i, m_i)$: If the entry $(\sigma_i, m_i, h_{2,i})$ exists in $\text{List}_{H_2}$, $\mathcal{C}$ returns $h_{2,i}$, otherwise, it chooses $h_{2,i} \leftarrow Z_q$, and inserts $(\sigma_i, m_i, h_{2,i})$ in $\text{List}_{H_2}$.

Queries to $H_3(K_i)$: If the entry $(K_i, h_{3,i})$ exists in $\text{List}_{H_3}$, $\mathcal{C}$ returns $h_{3,i}$, otherwise, it chooses $h_{3,i} \leftarrow \{0,1\}^n$, and inserts $(K_i, h_{3,i})$ in $\text{List}_{H_3}$.

Queries to $H_4(\sigma_i)$: If the entry $(\sigma_i, h_{4,i})$ exists in $\text{List}_{H_4}$, $\mathcal{C}$ returns $h_{4,i}$, otherwise, it chooses $h_{4,i} \leftarrow \{0,1\}^n$, and inserts $(\sigma_i, h_{4,i})$ in $\text{List}_{H_4}$.
Public key request: Upon receiving a public key request on \( ID_i \), \( C \) works as follows. If \( (\langle ID_i, U_i, Q_i \rangle, \zeta_i) \) exists in \( \text{List}_{pk} \), then it returns \( (ID_i, U_i, Q_i) \). Else, it flips a fair coin where \( \Pr[\zeta = 0] = \delta \), and works as follows (\( \delta \) will be determined later in the proof). If \( \zeta = 0 \), it runs the partial key extraction oracle below first, update \( \text{List}_{pk} \) and then output \( (ID_i, U_i, Q_i) \). If \( \zeta = 1 \), pick \( t \leftarrow \mathbb{Z}_q \), set \( Q_i \leftarrow aP \mod p \), adds \( (ID_i, U_i, \langle \bot, Q_i \rangle) \) to \( \text{List}_{\text{PartialSK}} \) and adds \( (\langle ID_i, U_i, Q_i \rangle, \zeta_i) \) to \( \text{List}_{pk} \), before outputting \( (ID_i, U_i, Q_i) \).

Partial key extraction: Upon receiving a partial key extraction query on \( (ID_i, U_i) \), \( C \) works as follow:

- If \( (ID_i, U_i, \langle w_i, Q_i \rangle) \in \text{List}_{\text{PartialSK}} \), return \( (w_i, Q_i) \).
- Else,
  - \( w_i \leftarrow \mathbb{Z}_q \), \( Z_i \leftarrow w_iP \mod p \), \( (j_1, \ldots, j_k) \leftarrow [1, \ldots, t] \), \( Q_i \leftarrow Z_i - \sum_{i=1}^{k} V_{j_i} + U_i \mod p \).
  - If \( (ID_i, Q_i, \ldots) \in \text{List}_{H_1} \), aborts. Else, adds \( (\langle ID_i, Q_i \rangle, h_{1,i}) \) to \( \text{List}_{H_1} \), where \( h_{1,i} \leftarrow (j_1, \ldots, j_k) \) and output the partial private key as \( (w_i, U_i, Q_i) \) after adding it to \( \text{List}_{\text{PartialSK}} \).

Secret key request: Upon receiving a secret key request on \( ID_i \), \( C \) checks if there exists a pair \( (ID_i, u_i, U_i) \in \text{List}_{\text{SecretKey}} \), it returns \( u_i \). Otherwise, selects \( u_i \leftarrow \mathbb{Z}_q \), computes \( U_i \leftarrow u_iP \mod p \) and inserts \( (ID_i, u_i, U_i) \) in \( \text{List}_{\text{SecretKey}} \).

Private key request: To answer a private key request on \( (ID_i, U_i) \), \( C \) runs the public key request oracle above to get \( (\langle ID_i, U_i, Q_i \rangle, \zeta_i) \in \text{List}_{pk} \) and finds \( (ID_i, u_i, U_i) \) in \( \text{List}_{\text{SecretKey}} \). If \( \zeta = 0 \), finds \( (ID_i, U_i, \langle w_i, Q_i \rangle) \in \text{List}_{\text{PartialSK}} \) and returns \( w_i + u_i \) as the response. Otherwise, it aborts.

Decryption query: Upon receiving a decryption query on \( (ID_i, Q_i, c_i = \langle R_i, u_i, v_i \rangle) \), \( C \) works as follows.

- Searches \( \text{List}_{pk} \) for an entry \( (\langle ID_i, U_i, Q_i \rangle, \zeta_i) \). If \( \zeta = 0 \), works as follows.
Searches $\text{List}_{\text{PartialSK}}$ for a tuple $(ID_i, U_i, \langle w_i, Q_i \rangle)$ and searches for $(ID, \langle w, Q \rangle)$ in $\text{List}_{\text{PartialSK}}$, set $\sigma' \leftarrow H_3((w + \alpha)R \mod p) \oplus u$, $m' = v \oplus H_4(\sigma')$, $r' := H_2(\sigma', m)$.

Checks if $R = r'P \mod p$ holds, outputs $m'$

- Else, if $\zeta = 1$, works as follows.

  - Runs the oracle for $H_1$ to get $h_{1,i}$ (to compute the public key $Y_i$) and checks lists $\text{List}_{H_2}, \text{List}_{H_3}$ and $\text{List}_{H_4}$ for tuples $(\langle \sigma_i, m_i \rangle, h_{2,i}), (K_i, h_{3,i})$, and $(\sigma_i, h_{4,i})$, such that $R_i = h_{2,i}P \mod p$, $u = h_{3,i} \oplus \sigma_i$ and $v = h_{4,i} \oplus m_i$ exists. Checks if $K_i = r_i(Y_i + Q_i)$ holds, outputs $m_i$, else, aborts.

After the first round of queries, $A_I$ outputs $ID^*$ and two messages $m_0$ and $m_1$ on which it wishes to be challenged on. We assume that $ID^*$ has been already queried to $H_1$ and was not submitted to the private key request oracle. $C$ checks $(\langle ID^*, U^*, Q^* \rangle, \zeta) \in \text{List}_{PK}$ if $\zeta = 0$, it aborts. Otherwise, it computes the challenge ciphertext as follows. $\beta^* \leftarrow \{0, 1\}, \sigma^* \leftarrow \{0, 1\}^n, u^* \leftarrow \{0, 1\}^n, b \leftarrow \{0, 1\}$. $R^* \leftarrow aP$ (this implicitly implies that $a = H_2(\sigma^*, m_b)$, $H_3(K_{ID^*}) \leftarrow u^* \oplus \sigma^*$ and $v^* \leftarrow H_4(\sigma^*) \oplus m_b$. Return $(R^*, u^*, v^*)$.

$A_I$ initiates the second round of queries similar as above, with the restrictions defined in Definition 15. When $A_I$ outputs its decision bit $b'$, $C$ returns a set $\Lambda = \{K_i - R_{id_i}^b, \text{where } K_i$s are the input queries to $H_3\}$.

Notice that if $C$ does not abort, and $A_I$ outputs its decision bit $b'$, then the public key must have the $Q_{ID^*} = aP$, and given how the challenge ciphertext is formed (e.g., $R^* = bP$), $K_{ID^*} = y_{ID^*}abP$ should hold, where $y_{ID^*}$ is known to $C$. Hence, the answer to a random instance of the the CDH problem $(P, aP, bP)$, can be derived from examining the $A_I$’s choice of public key and $H_3$ queries.
Here, we provide an indistinguishability argument for the above simulation. First we look at the simulation of the decryption algorithm. If $\zeta = 0$, we can see that the simulation is perfect.

For $\zeta = 1$, an error might occur in the event that $c_i$ is valid, but $(\sigma_i, m_i)$, $K_i$, and $\sigma_i$ were never queried to $H_2, H_3$, and $H_4$, respectively. For the first two hash functions, the probability that the $c_i$ is valid, given a query to $H_3$ was never made, considers the query to $H_2$ as well (not considering the checking phase in the simulation). Therefore, the probability that this could occur is $\frac{q_{H_2}}{2^n} + \frac{1}{p}$. When considering $H_4$, this probability is $\frac{1}{2^n} + \frac{1}{p}$. Given the number of decryption queries $q_D$, we have the probability of decryption error $\frac{q_D(q_{H_2}+1)}{2^n} + \frac{2q_{D}}{p}$.

$C$ will also fail in simulation during the partial key extraction queries if the entry $(ID_i, Q_i, \ldots)$ already exists in $\text{List}_{H_1}$. This will happen with probability $\frac{q_{H_1}}{2^n}$. The probability that $C$ does not abort in the simulation is $\delta^{q_{sk}}(1 - \delta)$ which is maximized at $\delta = 1 - \frac{1}{q_{sk}+1}$. Therefore, the probability that $C$ does not abort is $\frac{1}{e(q_{sk} + 1)}$, where $e$ is the base of natural logarithm. Given the argument above, we know that if $(\sigma^*, m_b)$, $(K^*)$ were never queried to $H_2$ and $H_3$ oracles, then $A_I$ cannot gain any distinguishing advantage more than $\frac{1}{2}$. Given all the above arguments, the probability that $K_{ID^*}$ has been queried to $H_3$ is $\geq \frac{2e}{e(q_{sk} + 1)} - \frac{q_{H_3}}{2^n} - \frac{q_D(q_{H_2}+1)}{2^n} - \frac{2q_{D}}{p}$.

Therefore, if the above probability occurs, $C$ can solve the CDH problem by finding and computing $K_{ID^*} = y_{ID^*}abP$ from the list $\Lambda$. Given the size of the list $\Lambda$ (i.e., $q_{H_3}$), the probability for $C$ to be successful in solving CDH is: $\epsilon' > \frac{1}{q_{H_3}} \left( \frac{2e}{e(q_{sk}+1)} - \frac{q_{H_3}}{2^n} - \frac{q_D(q_{H_2}+1)}{2^n} - \frac{2q_{D}}{p} \right)$.

**Theorem 3.** If an adversary $A_{II}$ can break the IND-CLE-CCA security of the encryption scheme proposed in Algorithm 4 after $q_{H_i}$ queries to random oracles $H_i$ for $i \in \{1, 2, 3, 4\}$, $q_D$ queries to the decryption oracle and $q_{sk}$ to the secret key extraction oracle with probability $\epsilon$. There exists another algorithm $C$ that runs $A_{II}$ as subroutine and breaks a random instance of the CDH problem $(P, aP, bP)$ with probability $\epsilon'$ where: $\epsilon' > \frac{1}{q_{H_3}} \left( \frac{2e}{e(q_{sk}+1)} - \frac{q_{H_3}}{2^n} - \frac{q_D(q_{H_2}+1)}{2^n} - \frac{2q_{D}}{p} \right)$. 

44
Sketch. Having access to random oracles, and by keeping lists similar to above, the challenger \( C \) can simulate an indistinguishable environment for \( A_{II} \) and respond to its queries similar to the above proof. Note that following Definition 17, \( A_{II} \) can query for the secret key of all the users, except for the target user \( ID^* \).

\( C \) knows the \( t \) private values \( v'_i \)'s in the scheme, and tries to embed a random instance of the CDH problem \((P,aP,bP)\). By flipping a fair coin, as in the public key request query above, \( C \) defines the probability to embed \( aP \) in the target \( U_{ID^*} \) value. \( C \) sets \( bP \) as a part of the challenge ciphertext (i.e., \( R^* \leftarrow bP \)).

After the \( A_{II} \) outputs a forgery, \( C \) can extract the solution to the CDH problem since it has knowledge over the \( t \) secret values and \( \beta^* \).

\[ \square \]

Lemma 2. A public key replacement attack by \( A_I \) is not practical since it will falsify the private key.

**Proof.** Note that if \( A_I \) replaces \( U \) with a new value \( U' \) (which it might know the corresponding secret key), then the existing \( Q \) will be falsified since \( Q = U + W \) and this will also falsify the current partial private key component \( y \) since it is computed based on the indexes that are obtained by computing \( H_1(ID, Q) \). Also note that, if \( A_I \) can obtain the original \((\alpha, U)\), given \( Q \) is public, it can compute \( W \), however, \( W \) is merely the commitment of \( \beta \) and it does not disclose any information about \( \beta \). The public key replacement attack in our security proof is possible if \( A_I \) requests a new partial private key for each new \( U' \).

\[ \square \]

Lemma 3. If an adversary \( A_I \) can break the EU-CMA security of the signature scheme proposed in Algorithm 5, then one can build another algorithm \( C \) that runs \( A_I \) as subroutine and breaks a random instance of the ECDLP \((P,aP)\).
Proof. Due to the space constraint, here we give the high level idea of our proof. We let \( A_I \) be as in Definition 18, then we can build another algorithm \( C \) that uses \( A_I \) as a subroutine, and upon \( A_I \)'s successful forgery, solves a random instance of the ECDLP \((P, aP)\). \( C \) knows the \( t \) secret values \( v_i \)'s and, similar to the proof of Theorem 2, it sets \( aP \) as a part of the target user’s public key (i.e., \( Q_{ID^*} \leftarrow aP \)). Most of the simulation steps are like the ones in the proof of Theorem 2. At the end of the simulation phase, \( A_I \) outputs a forgery signature \((s_1^*, e_1^*)\), the proof then uses the forking lemma [173] to run the adversary again to obtain a second forgery \((s_2^*, e_2^*)\), using the same random tape. Our proof will follow the same approach as in [99] which is very similar to the proof in [182]. Given two forgeries and the knowledge of \( C \) on the \( v'_i \)'s and \( \alpha_{ID^*} \), \( C \) can compute \( a \) and solve the ECDLP. Note that similar to Schnorr [182] the security of the scheme will be non-tight due to the forking lemma.

4.4.1 Parameters Selection for \((t, k)\)

Parameters \((t, k)\) should be selected such that the probability \( q_{H_1} \frac{k!}{2^\gamma} \) is negligible. Considering that \( \gamma = k \log_2 t \) (since \( k \) indexes that are \( \log_2 t \)-bit long are selected with the hash output), this gives us \( q_{H_1} \frac{k!}{2^{k|t|}} \). We further elaborate on some choices of \((t, k)\) along with their performance implications in §4.5.

4.5 Performance Analysis and Comparison

We first present the analytical and then experimental performance analysis and comparison of our schemes with their counterparts. We focus on the online operations (e.g., encryption, signing, key exchange) for which both our IDB and CL schemes have the same algorithms, rather than one-time (offline) processes like setup and key generation. Since the online operations are identical in IDB and CL systems in our case, we refer to them as “Our Schemes” in the following tables/discussions.
Table 4.1: Analytical comparison of public key encryption schemes

<table>
<thead>
<tr>
<th>Scheme</th>
<th>sk</th>
<th>Enc</th>
<th>Comm.</th>
<th>Dec</th>
<th>pk</th>
<th>mpk</th>
<th>System Type</th>
<th>$\kappa$</th>
</tr>
</thead>
<tbody>
<tr>
<td>ECIES [189]</td>
<td>$</td>
<td>q</td>
<td>$</td>
<td>$2m_{EC} + dm_{EC}$</td>
<td>$2</td>
<td>p</td>
<td>+ b + \frac{d}{d + CF}$</td>
<td>$m_{EC}$</td>
</tr>
<tr>
<td>BF [53]</td>
<td>$</td>
<td>p</td>
<td>$</td>
<td>$bp + m_{EC} + ex$</td>
<td>$</td>
<td>p</td>
<td>+ b +</td>
<td>M</td>
</tr>
<tr>
<td>AP [8]</td>
<td>$</td>
<td>p</td>
<td>$</td>
<td>$3bp + m_{EC} + ex$</td>
<td>$</td>
<td>p</td>
<td>+ b +</td>
<td>M</td>
</tr>
<tr>
<td>BSS [22]</td>
<td>$2</td>
<td>q</td>
<td>$</td>
<td>$4ex + m$</td>
<td>$</td>
<td>p</td>
<td>+ b +</td>
<td>M</td>
</tr>
<tr>
<td>WSB [208]</td>
<td>$</td>
<td>p</td>
<td>+</td>
<td>q</td>
<td>$</td>
<td>$3m_{EC} + 2a_{EC}$</td>
<td>$2</td>
<td>q</td>
</tr>
<tr>
<td>Our Schemes</td>
<td>$</td>
<td>p</td>
<td>$</td>
<td>$2m_{EC} + ka_{EC}$</td>
<td>$</td>
<td>p</td>
<td>+ b +</td>
<td>M</td>
</tr>
</tbody>
</table>

$\dagger$ Denotes the security bit. Enc, Dec, and Comm. represent encryption, decryption, and communication load (bi-directional), respectively. $m_{EC}, a_{EC},$ and $dm_{EC}$ denote the costs of EC scalar multiplication, EC addition, and double scalar multiplication over modulus $p$, respectively. $m, ex$ and $bp$ denote multiplication, exponentiation and pairing operation, respectively. $k$ is the BPV parameter that shows how many precomputed pairs are selected in the online phase. $b, d$ and $CF$ denote block/key size for symmetric key encryption, message digest (i.e., MAC) size and size of the certificate, respectively. $M$ denotes message space size. TD, IDB, and CL represent traditional public key cryptography, identity-based cryptography, and certificateless cryptography, respectively.

Table 4.2: Analytical comparison of digital signature schemes

<table>
<thead>
<tr>
<th>Scheme</th>
<th>sk</th>
<th>Sign</th>
<th>Comm.</th>
<th>Verify</th>
<th>pk</th>
<th>mpk</th>
<th>System Type</th>
<th>$\kappa$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Schnorr [182]</td>
<td>$</td>
<td>q</td>
<td>$</td>
<td>$m_{EC}$</td>
<td>$2</td>
<td>q</td>
<td>+ CF$</td>
<td>$2dm_{EC}$</td>
</tr>
<tr>
<td>GG [99]</td>
<td>$</td>
<td>q</td>
<td>$</td>
<td>$m_{EC}$</td>
<td>$2</td>
<td>p</td>
<td>+</td>
<td>q</td>
</tr>
<tr>
<td>AP [8]</td>
<td>$</td>
<td>p</td>
<td>$</td>
<td>$2m_{EC} + a_{EC} + bp$</td>
<td>$</td>
<td>q</td>
<td>+</td>
<td>p</td>
</tr>
<tr>
<td>KIB [128]</td>
<td>$</td>
<td>q</td>
<td>$</td>
<td>$m_{EC}$</td>
<td>$</td>
<td>q</td>
<td>+</td>
<td>p</td>
</tr>
<tr>
<td>Our Schemes</td>
<td>$</td>
<td>q</td>
<td>$</td>
<td>$m_{EC}$</td>
<td>$2</td>
<td>q</td>
<td>$</td>
<td>$dm_{EC} + ka_{EC}$</td>
</tr>
</tbody>
</table>

We consider the cost of certificate verification for schemes in traditional PKI. We only consider the cost of verifying and communicating the cost of one certificate, which is highly conservative since in practice (i.e., X.509) there are at minimum two certificates in a certificate chain. This number could be as high as ten certificates in some scenarios.

4.5.1 Analytical Performance Analysis and Comparison

We present a detailed analytical performance comparison of our schemes with their counterparts for public key encryption/decryption, digital signature and key exchange in Table 4.1, Table 4.2 and Table 4.3, respectively.

Our schemes have significantly lower communication overhead than their PKI-based counterpart in all cryptosystems as they do not require the transmission of certificates. As discussed above and also elaborated in Section 4.5.2, this translates into substantial bandwidth gain as well
Table 4.3: Analytical comparison of key exchange schemes

<table>
<thead>
<tr>
<th>Scheme</th>
<th>sk</th>
<th>User Comp.</th>
<th>Comm.</th>
<th>pk</th>
<th>mpk</th>
<th>System Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ephemeral ECDH</td>
<td>q</td>
<td>2m_EC</td>
<td>2</td>
<td>p</td>
<td>+ CF</td>
<td>p</td>
</tr>
<tr>
<td>ECHMQV [133]</td>
<td>q</td>
<td>3m_EC</td>
<td>2</td>
<td>p</td>
<td>+ CF</td>
<td>p</td>
</tr>
<tr>
<td>TFNS [198]</td>
<td>p</td>
<td>bp + 5m_EC</td>
<td></td>
<td>p</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AP [8]</td>
<td>p</td>
<td>4bp + ex</td>
<td>3</td>
<td>p</td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>YT [211]</td>
<td>2</td>
<td>q</td>
<td>+</td>
<td>p</td>
<td>+</td>
<td>s</td>
</tr>
<tr>
<td>Our Scheme</td>
<td>q</td>
<td>3m_EC + (k + 1)a_EC</td>
<td>2</td>
<td>p</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

User Comp. denotes user computation.
†|s| denotes the output of a signature scheme that the authors in [211] use in their scheme.

Table 4.4: Public key encryption schemes on commodity hardware

<table>
<thead>
<tr>
<th>Scheme</th>
<th>sk</th>
<th>Enc</th>
<th>Comm.</th>
<th>Dec</th>
<th>pk</th>
<th>mpk</th>
<th>E2E Delay</th>
</tr>
</thead>
<tbody>
<tr>
<td>ECIES [189]</td>
<td>32</td>
<td>55</td>
<td>690 +</td>
<td>M</td>
<td></td>
<td>21</td>
<td>32</td>
</tr>
<tr>
<td>BF [53]</td>
<td>32</td>
<td>≈ 2000</td>
<td>48 +</td>
<td>M</td>
<td></td>
<td>≈ 2000</td>
<td>32</td>
</tr>
<tr>
<td>AP [8]</td>
<td>32</td>
<td>≈ 6000</td>
<td>48 +</td>
<td>M</td>
<td></td>
<td>≈ 2000</td>
<td>64</td>
</tr>
<tr>
<td>BSS [22]</td>
<td>64</td>
<td>73</td>
<td>64 +</td>
<td>M</td>
<td></td>
<td>53</td>
<td>64</td>
</tr>
<tr>
<td>WSB [208]</td>
<td>64</td>
<td>53</td>
<td>64 +</td>
<td>M</td>
<td></td>
<td>41</td>
<td>64</td>
</tr>
<tr>
<td>Our Schemes</td>
<td>32</td>
<td>39</td>
<td>48 +</td>
<td>M</td>
<td></td>
<td>33</td>
<td>32</td>
</tr>
</tbody>
</table>

All sizes are in Bytes, and all computations are in microseconds.
† We assume the certificate size is 578 Bytes, the size is given in RFC 5280 [77].

as computational efficiency since the certification verification overhead is also lifted. Moreover, in almost all instances, our schemes also offer a lower end-to-end computational overhead compared to their PKI-based counterparts. Our schemes also offer a lower end-to-end computational delay than that of all of their IDB and CL counterparts in all cryptosystems, with generally equal private and public key sizes. However, the master public key size of our scheme is larger than all of their counterparts.

4.5.2 Experimental Performance Analysis and Comparison

We now further elaborate on the details of our performance analysis and comparison with experimental results. We conduct experiments on both commodity hardware and low-end embedded devices that are typically found in IoT systems to objectively assess the performance of our schemes as well as their counterparts. Our open-sourced implementation is available via the following link.

https://github.com/Rbehnia/CertFreeSystems
Table 4.5: Digital signature schemes on commodity hardware

<table>
<thead>
<tr>
<th>Scheme</th>
<th>sk</th>
<th>Sign</th>
<th>Comm.</th>
<th>Verify</th>
<th>pk</th>
<th>mpk</th>
<th>E2E Delay</th>
</tr>
</thead>
<tbody>
<tr>
<td>Schnorr [182]</td>
<td>32</td>
<td>12</td>
<td>642</td>
<td>44</td>
<td>32</td>
<td>-</td>
<td>56</td>
</tr>
<tr>
<td>GG [99]</td>
<td>32</td>
<td>12</td>
<td>96</td>
<td>44</td>
<td>32</td>
<td>32</td>
<td>56</td>
</tr>
<tr>
<td>AP [8]</td>
<td>32</td>
<td>≈2000</td>
<td>64</td>
<td>≈8000</td>
<td>64</td>
<td>32</td>
<td>≈10000</td>
</tr>
<tr>
<td>KIB [128]</td>
<td>32</td>
<td>20</td>
<td>64</td>
<td>61</td>
<td>96</td>
<td>64</td>
<td>81</td>
</tr>
<tr>
<td>Our Schemes</td>
<td>32</td>
<td>12</td>
<td>64</td>
<td>27</td>
<td>32</td>
<td>32K</td>
<td>39</td>
</tr>
</tbody>
</table>

All sizes are in Bytes, and all computations are in microseconds.

Table 4.6: Key exchange schemes on commodity hardware

<table>
<thead>
<tr>
<th>Scheme</th>
<th>sk</th>
<th>User Comp.</th>
<th>Comm.</th>
<th>pk</th>
<th>mpk</th>
<th>E2E Delay</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ephemeral ECDH [82]</td>
<td>32</td>
<td>55</td>
<td>642</td>
<td>32</td>
<td>-</td>
<td>110</td>
</tr>
<tr>
<td>ECHMQV [133]</td>
<td>32</td>
<td>74</td>
<td>642</td>
<td>32</td>
<td>-</td>
<td>148</td>
</tr>
<tr>
<td>TFNS [198]</td>
<td>32</td>
<td>≈2000</td>
<td>32</td>
<td>32</td>
<td>32</td>
<td>≈4000</td>
</tr>
<tr>
<td>AP [8]</td>
<td>32</td>
<td>≈8000</td>
<td>96</td>
<td>64</td>
<td>32</td>
<td>≈16000</td>
</tr>
<tr>
<td>YT† [211]</td>
<td>160</td>
<td>157</td>
<td>160</td>
<td>128</td>
<td>64</td>
<td>314</td>
</tr>
<tr>
<td>Our Scheme</td>
<td>32</td>
<td>57</td>
<td>64</td>
<td>32</td>
<td>32K</td>
<td>114</td>
</tr>
</tbody>
</table>

All sizes are in Bytes, and all computations are in microseconds.
† The signature size is considered as 64 bytes.

4.5.2.1 Experiments on Commodity Hardware:

We used an i7 Skylake laptop equipped with a 2.6 GHz CPU and 12 GB RAM in our experiments. We implemented our schemes on the FourQ curve [78] which offers fast elliptic curve operations for $\kappa = 128$-bit security. We instantiated our random oracles with blake2 hash function\(^6\), which offers high efficiency and security. For our parameters, we selected $k = 18$ and $t = 1024$. We conservatively estimated the costs of our counterparts based on the microbenchmarks on our evaluation setup of (i) FourQ curve for schemes that do not require pairing and (ii) PBC library\(^7\) on a curve with $\kappa = 80$-bit security (we used the most efficient alternative for them) for schemes that require pairing.

As depicted in Table 4.4, the encryption and decryption algorithms of our schemes are more efficient than their counterparts in the identity-based and certificateless settings. More specifically, the end-to-end delay of our schemes is $\approx 25\%$ lower than that in [208], which is specifically suitable

\(^6\)http://131002.net/blake/blake.pdf
\(^7\)https://crypto.stanford.edu/pbc/
Table 4.7: Public key encryption schemes on 8-bit AVR processor

<table>
<thead>
<tr>
<th>Scheme</th>
<th>$sk$</th>
<th>$Enc$</th>
<th>Comm.</th>
<th>$Dec$</th>
<th>$pk$</th>
<th>$mpk$</th>
</tr>
</thead>
<tbody>
<tr>
<td>ECIES [189]</td>
<td>32</td>
<td>17 950 967</td>
<td>610 + $</td>
<td>c</td>
<td>$</td>
<td>6 875 861</td>
</tr>
<tr>
<td>BF [53]</td>
<td>32</td>
<td>60 802 555</td>
<td>48 + $</td>
<td>c</td>
<td>$</td>
<td>56 278 012</td>
</tr>
<tr>
<td>AP [8]</td>
<td>32</td>
<td>166 213 087</td>
<td>48 + $</td>
<td>c</td>
<td>$</td>
<td>58 912 609</td>
</tr>
<tr>
<td>BSS [22]</td>
<td>64</td>
<td>22 791 835</td>
<td>64 + $</td>
<td>M</td>
<td>$</td>
<td>16 590 321</td>
</tr>
<tr>
<td>WSB [208]</td>
<td>64</td>
<td>17 091 636</td>
<td>64 + $</td>
<td>c</td>
<td>$</td>
<td>13 631 755</td>
</tr>
<tr>
<td>Our Schemes</td>
<td>32</td>
<td>11 789 632</td>
<td>48 + $</td>
<td>c</td>
<td>$</td>
<td>9 883 161</td>
</tr>
</tbody>
</table>

All sizes are in Bytes, and all computations are in CPU Cycles.

Table 4.8: Digital signature schemes on 8-bit AVR processor

<table>
<thead>
<tr>
<th>Scheme</th>
<th>$sk$</th>
<th>$Sign$</th>
<th>Comm.</th>
<th>$Verify$</th>
<th>$pk$</th>
<th>$mpk$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Schnorr [182]</td>
<td>32</td>
<td>4 263 298</td>
<td>642</td>
<td>17 902 958</td>
<td>32</td>
<td>-</td>
</tr>
<tr>
<td>GG [99]</td>
<td>32</td>
<td>4 263 298</td>
<td>96</td>
<td>17 902 958</td>
<td>32</td>
<td>32</td>
</tr>
<tr>
<td>AP [8]</td>
<td>32</td>
<td>62 487 032</td>
<td>64</td>
<td>221 226 015</td>
<td>64</td>
<td>32</td>
</tr>
<tr>
<td>KIB [128]</td>
<td>32</td>
<td>7 025 861</td>
<td>64</td>
<td>20 617 583</td>
<td>96</td>
<td>64</td>
</tr>
<tr>
<td>Our Schemes</td>
<td>32</td>
<td>4 263 298</td>
<td>64</td>
<td>10 955 369</td>
<td>32</td>
<td>32K</td>
</tr>
</tbody>
</table>

All sizes are in Bytes, and all computations are in CPU Cycles.

for aerial drones. One could also notice how the communication overhead is lower in certificateless and identity-based schemes since there is no need for certificate transmission.

As shown in Table 4.5, our schemes enjoy from the fastest verification algorithms among all its counterparts. This is again due to the novel way the user keys are derived and results in 30% and 52% faster end-to-end delay as compared to its most efficient identity-based [99] and certificateless [128] counterparts, respectively. One may notice that although schemes in [99, 128, 182], along with our schemes, all require a scalar multiplication in their signature generation (see Table 4.2), their experimental costs differ. The reason for this discrepancy is the fact that the cost of scalar multiplication over the generator $P$ is faster than the scalar multiplication over any curve points, and these differences are considered in the experimental evaluations.

As shown in Table 4.6, our schemes’ performance is similar to that in their counterparts in traditional PKI setting [82, 133]. However, they outperform the most efficient counterpart in certificateless setting [211] by having 65% lower end-to-end delay and 60% smaller load for communication.
Table 4.9: Key exchange schemes on 8-bit AVR processor

<table>
<thead>
<tr>
<th>Scheme</th>
<th>sk</th>
<th>User Comp.</th>
<th>Comm.</th>
<th>pk</th>
<th>mpk</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ephemeral ECDH</td>
<td>32</td>
<td>18 039 710</td>
<td>642</td>
<td>32</td>
<td>-</td>
</tr>
<tr>
<td>ECHMQV</td>
<td>32</td>
<td>24 601 857</td>
<td>642</td>
<td>32</td>
<td>-</td>
</tr>
<tr>
<td>TFNS</td>
<td>32</td>
<td>82 356 781</td>
<td>32</td>
<td>32</td>
<td>32</td>
</tr>
<tr>
<td>AP</td>
<td>32</td>
<td>221 226 015</td>
<td>96</td>
<td>64</td>
<td>32</td>
</tr>
<tr>
<td>YT [211]</td>
<td>160</td>
<td>54 212 824</td>
<td>160</td>
<td>128</td>
<td>64</td>
</tr>
<tr>
<td>Our Scheme</td>
<td>32</td>
<td>18 015 493</td>
<td>64</td>
<td>32</td>
<td>32K</td>
</tr>
</tbody>
</table>

All sizes are in Bytes, and all computations are in CPU Cycles.

4.5.2.2 Experiments on Low-End Device

We used an 8-bit AVR ATmega 2560 microprocessor to evaluate the costs of our schemes on an IoT device. AVR ATmega 2560 is a low-power microprocessor with 256 KB flash, 8 KB SRAM, 4 KB EEPROM, and operates at 16 MHz frequency. We used the 8-bit AVR library of the FourQ curve presented in [142]. For our counterparts, we again conservatively estimated their costs based on microbenchmarks in (i) FourQ curve 8-bit AVR implementation [142], and (ii) NanoECC [195], that implements a curve that supports pairings on 8-bit AVR microprocessors and offers $\kappa = 80$-bit security.

As depicted in Table 4.7, our schemes outperform all of their identity-based and certificateless counterparts and have a more efficient encryption algorithm than [189]. Our decryption algorithms, while being more efficient than all of their identity-based and certificateless counterparts, are slightly less efficient than the one in [189]. Similar to the trend in the analytical performance, our signature schemes outperform their counterparts. As Table 4.8 shows, our schemes’ signing algorithm are amongst the most efficient ones, while the verification algorithm outperforms all the counterparts with similar communication overhead.

4.5.3 Limitations

The main limitation of our schemes is the size of the master public key. Note that if there are different TTP in different domains and users often communicate with the users in those domains,
it would make sense to store different $mpk$. Otherwise, the users only need to store $mpk$ for their own systems. We can reduce the size of the $mpk$ in exchange for a small performance loss. For instance, with $k = 32$, we can reduce the size of the $mpk$ by four times.

4.6 Related Work

There is a comprehensive literature covering different aspects of IDB and CL systems. Remark that most of the closely related works have been discussed in Section 3 and 5 in terms of security models and performance metrics. Overall, the main difference in our work is to focus on the achievement of inter-compatibility between IDB and CL with a high efficiency, with respect to existing alternatives.

The idea of IDB cryptography was proposed by Shamir [186]. However, the first practical instance of such schemes was proposed later by Boneh and Franklin [53] using bilinear pairing. To get the full adaptive-identity, chosen-ciphertext security guarantees without sacrificing performance, Boyen [61] described an augmented versions of the scheme in [51] in the random-oracle model. However, the augmented version also requires multiple pairing computations in the decryption algorithm. Following [37], several pairing-free signature scheme were proposed. Galindo and Garcia [99] proposed a lightweight IDB signature scheme based on [182] with reduction to the discrete logarithm problem.

CL cryptography [8] was proposed to address the private key escrow problem in IDB systems. In the same paper, the authors proposed an IND-CCA encryption scheme along with a signature and key exchange schemes. Following their work, Baek et al. [22] proposed the first IND-CCA secure certificateless encryption scheme without pairing. The scheme is constructed using Schnorr-like signatures in partial private key generation algorithm. Recently Won et al. [208] proposed another efficient IND-CCA encryption scheme that is specifically used for key encapsulation mechanisms.
There has been a number of works that focus on the security models of certificateless systems. In most of the proposed models (e.g., [8]) a Type-II adversary is assumed to generate the keys honestly, and initiate the attacks only after the setup phase.
Chapter 5: Fast Signatures from Compact Knapsack

5.1 Introduction

Ever since Shor [188] published polynomial-time quantum algorithms for factoring and discrete logarithm, the threat of quantum computation has loomed ominously over public-key cryptography. Since traditional public-key cryptography is broken by quantum attacks, alternative schemes with post-quantum (PQ) security must be identified before quantum computers become practical.

Recently, the NSA has announced an advisory on the possibility of transitioning to PQ-secure cryptography in the near future [160]. To avoid a hasty transition from current conventional cryptosystems to PQ-secure systems, NIST has already initiated the first round of standardizations for PQ cryptography.

5.1.1 The State of the Art and Limitations

Lamport [138] proposed the first PQ-secure one-time signature scheme based on the idea of committing to secret keys via one-way functions. Later, Bos and Chaum [57] and Reyzin and Reyzin [179] proposed different variants of Lamport’s signature with the aim of minimizing the public key and signature size, respectively. Today, digital signatures based on lattices, hash functions, codes, multivariates and symmetric primitives are the leading practical candidates with PQ security.

- **Lattice-based Signature:** There are two main categories of lattice-based signature schemes. One is focusing on hardness of worst-case to average-case problems with standard lattices (e.g.,
While they provide a strong security, they suffer from very large parameter sizes (in the orders of a few MBs). Another direction, with more focus on efficiency, is based on ring analogs of standard lattice problems (e.g., [7, 85, 87]). Most of these efficient schemes, however, suffer from costly sampling operations with high precision over some normal distribution (e.g., Gaussian sampling) during the signing. Relaxation of this requirement, by only sampling over integers, permitted more efficient constructions like BLISS [85], which is based on the Fiat-Shamir transform [96]. Later, Ducas et al. proposed a hash-and-sign signature scheme [89] that has a smaller signature and key size than BLISS, but with slower signing due to expensive discrete Gaussian sampling over a lattice.

Gaussian sampling not only incurs a performance penalty, but its implementation is also prone to side-channel attacks. For instance, BLISS [85] has been targeted with a number of side-channel attacks [94, 112]. At the moment, avoiding such side channels in implementation is considered to be highly challenging and error-prone [87]. Recently, Ducas et al. proposed a new Fiat-Shamir-based scheme called Dilithium [87], which avoids Gaussian sampling during signing. The security of Dilithium is based on the learning with errors (LWE) and short integer solution (SIS) problems in ideal lattices.

qTESLA [48] is another lattice-based signature scheme proposed to the first round of NIST standardization for PQ cryptography. qTESLA is based on the decisional ring learning with errors (R-LWE) problem. While similar to Dilithium [87], qTESLA avoids using Gaussian Sampling during signature generation, but it suffers from a higher end-to-end delay.

pqNTRUSign [120] is an instantiation of modular lattice signature (over the NTRU lattice). Signatures can be generated using a (bimodal) Gaussian or a uniform sampler. Similar to Dilithium [87], pqNTRUSign employs rejection sampling to avoid the leakage of the private
key components. However, with the current suggested parameters, the scheme suffers from a high signing time that is due to the high rejection rate.

While other lattice-based primitives, such as key-exchange protocols, have undergone some real-world testing and evaluations (e.g., [64]), the current precarious state of lattice-based approaches has hindered the development of PQ-secure signatures.

- **Hash-Based Signatures**: Hash-based signatures can be proven secure in the standard model under the very well-studied properties of hash functions such as pre-image resistance. The combination of Merkle trees [153] with early one-time hash-based signatures (e.g., Lamport [138]) results in very efficient stateful schemes which are secure for a number of signatures. Traditional hash-based schemes are *stateful*, to ensure that the signer does not reuse some of the private key materials. Recently, stateless signatures (e.g., SPHINCS [47]) have been proposed. SPHINCS has a tight security reduction to the security of its building blocks such as hash functions and PRNGs. Unfortunately, these schemes have large signatures ($\approx 41$ KB) and very costly signature generation, especially on low-end devices [124].

- **Code-Based Signatures**: Code-based cryptography has been largely affected by the Syndrome Decoding Problem [42]. Since McEliece cryptosystem [152], which is based on binary Goppa codes, there have been a lot of efforts in balancing security and efficiency of such systems. The most well-studied and provably secure approach to obtain signature schemes is applying the Fiat-Shamir transform [96] on the identification scheme proposed by Véron [203] and Stern [193]. pqsigRM [139] is a new code-based signature scheme based on punctured Reed-Muller (RM) submitted to the first NIST post-quantum standardization conference. pqsigRM can be considered as a highly improved version of the scheme in [80], where most of the improvements are due to the replacement of Goppa Codes in [80] with punctured RM codes. While pqsigRM
has significantly improved the overall parameters sizes in [80], the key sizes are still larger than its lattice-based and hash-based counterparts.

- **Multivariate-Based Signatures**: There are a number of multivariate-based signatures submitted to the NIST standardization of PQ cryptography. For instance, GeMSS [69] can be considered as an improvement of its predecessor QUARTZ [168], that is based on the Hidden Field Equations cryptosystems. GeMSS enjoys from an efficient verification algorithm and very compact signatures, however, the signing algorithm is significantly slower than its hash-based counterparts (e.g., SPHINCS+ [123]).

- **Symmetric Key Based Signatures**: PICNIC [71] is another novel construction which is based on the problems related to symmetric key cryptography. PICNIC is obtained by applying the Fiat-Shamir transform on an efficient zero-knowledge proof which results in very short public key and private key sizes. However, the scheme suffers from large signature sizes with relatively slow (as compared to lattice-based schemes) signing and verification algorithms.

5.1.2 Our Contribution

We propose a simple and efficient PQ-secure signature scheme, **TACHYON**, based on well-studied primitives. We outline a comparison between **TACHYON** and some of its other PQ-secure counterparts in Table 5.2 (see §5.5), and further elaborate on its desirable properties below:

- **New Algorithmic Design**: **TACHYON** can be viewed as a novel modification of the HORS construction [179], which is based on one-way functions. We harness the HORS approach with the generalized compact knapsack (GCK) of Micciancio [154]. The additively homomorphic property of GCK provides two benefits: It allows us to compress the signature size as compared to one-time signatures, and more importantly, it leads to a totally new paradigm for extending
few-time hash-based signatures to *stateless* schemes supporting polynomially-bounded number of signatures.

The security of our scheme is based on the one-wayness of GCK function family. These properties reduce to the worst-case hardness of problems in cyclic lattices [147, 154].

- **Improved Side-Channel Resiliency**: It has been shown that Gaussian sampling is prone to side-channel attacks (e.g., [112, 172]). Since side channels are a property of an algorithm’s implementation, they can be somewhat mitigated with suitable implementation techniques. However, the process of eliminating side channels in Gaussian sampling algorithms (e.g., in BLISS [85]) is known to be arduous and error-prone [87]. TACHYON does not require any variants of Gaussian sampling. Instead, it uses uniform sampling over a bounded domain, and rejection sampling to check for an outputted signature to be in a safe range.

- **Fast Verification**: The verification algorithm of TACHYON is very efficient, involving only two hash function calls, a GCK one-way function call, and vector additions. This makes TACHYON the most verifier computationally efficient alternative among its counterparts. For example, using TACHYON with 256-bit security, it is possible to verify 35,714 messages per second on commodity hardware (e.g., Intel 6th generation i7 processor), which is up to 3.7× faster than Dilithium [87], one of its fastest alternatives.

- **Fast Signing**: Signature generation of TACHYON does not require any costly operations (e.g., Gaussian sampling) but only a GCK function call (which is demonstrated to be fast [148]), along with a small constant number of pseudorandom function (PRF) calls and a small number of vector additions. This makes the signature generation of TACHYON the fastest as compared to its counterparts.
• **Small Private Key:** The private keys in TACHYON are as small as $\kappa$-bit, which is the smallest among existing PQ-secure schemes. Furthermore, unlike some other schemes (e.g., [85]), the signer does not need to store a pre-computed table to be used in the sampling process. Along with the signer computational efficiency, this property makes TACHYON a feasible alternative for low-end devices.

• **Tunable Parameters:** Our new algorithmic design allows us to offer various speed and storage trade-offs based on the parameter choices. For instance, one can pre-compute and store some intermediate values at the signer’s side in exchange for a faster signing, reduce the public key and/or signature size but with an increase in the end-to-end delay, or increase the signature size to offer lower rejection sampling rates for a faster signing. Some of these possible trade-offs are further elaborated in §5.5.2.

5.1.3 Limitations

All of these desirable properties of TACHYON come at the cost of a larger public key. For instance, the public key in TACHYON-256 is as large as 2976 KB, whereas it is only 1760 bytes in Dilithium[87]. Yet, we believe there are many use-cases where storing a larger public key is tolerable. For instance, a resourceful command center that verifies a large number of signatures from sensors can store such a public key. However, if the verifier is strictly memory-limited and cannot afford to store large public keys, then schemes with a smaller public key, such as Dilithium, should be considered.
5.2 Building Blocks

Since TACHYON is inspired by the construction of Bos and Chaum (BC) signature scheme which uses a bijective function $S(\cdot)$ and a one-way function (OWF) $f(\cdot)$ [57], we briefly explain about a simple generalization of their construction in the following definition.

Definition 19. BC signature scheme consists of three algorithms $\text{BC} = (\text{Kg}, \text{Sig}, \text{Ver})$ defined as follow.

- $(sk, pk) \leftarrow \text{BC.Kg}(1^n)$ Given the security parameter $1^n$ it sets $t, k$ and $l$ and generate $t$ random $l$--bit values for the private key $(x_1, \ldots, x_k)$ and compute the public key components $(y_i)$ as the image of the private key components $x_i$ with respect to a one-way function $f(\cdot)$, i.e., $y_i \leftarrow f(x_i)$ where $i \in \{1, \ldots, t\}$. Finally set $sk \leftarrow (x_1, \ldots, x_t)$ and $pk \leftarrow (t, k, \langle y_1, \ldots, y_t \rangle)$.

- $\sigma \leftarrow \text{BC.Sig}(M, sk)$: Given a b-bit message $M$ and $sk$, interpret $M$ as an integer between 0 and $2^b - 1$ and set $(i_1, \ldots, i_k)$ as the $M$--th $k$--element subset of set $(1, 2, \ldots, t)$, computed as $S(M)$. Output the signature as $\sigma \leftarrow (x_{i_1}, \ldots, x_{i_k})$.

- $\{0,1\} \leftarrow \text{BC.Ver}(M, \sigma, pk)$: Given a message-signature pair $(M, \sigma = (x'_1, x'_2, \ldots, x'_t))$, interpret $M$ as an integer between 0 and $2^b - 1$ and set $(i_1, \ldots, i_k)$ as the $M$--th $k$--element subset of set $(1, 2, \ldots, t)$, computed as $S(M)$. It the checks if $\{y_{ij} = f(x'_{ij})\}_{j=1}^{t}$ holds, it outputs 1, else it outputs 0.

5.3 Proposed Scheme

5.3.1 TACHYON

Our conceptual starting point is the HORS construction [179], which itself is a variant of the Bos and Chaum scheme [57]. The private key consists of many random values $x_i$, and the public key consists of corresponding images $y_i = F(x_i)$, where $F$ is a one-way function. Of course, the $x_i$
values can be derived from a small seed using a PRF (this feature is preserved by TACHYON, and leads to a minimal signing key). To sign a message $M$, the signer first computes $H_2(M)$ and interprets it as a sequence of indices $(i_1, \ldots, i_k)$. The signature then consists of $x_{i_1}, \ldots, x_{i_k}$. To verify, one can simply compare $F(x_j)$ to the public key value $y_j$, for each relevant $j$.

Our novel departure from this paradigm is to use an additively homomorphic OWF $F$. Specifically, we choose the generalized compact knapsack (GCK) function family of Micciancio [154]. This allows the signature to be compressed, as follows. Instead of $x_{i_1}, \ldots, x_{i_k}$, the signature can contain only $s = \sum_j x_{i_j}$. The verifier can then check that $F(s) = \sum_j y_{i_j}$.

However, this approach leaks a linear combination of the secret key material. After a moderate number of signatures, it would be possible to solve for the entire secret key via a system of linear equations. To thwart this, we add some “noise”. Specifically, the signature consists of $s = \sum_j x_{i_j} + r'$ for a suitably distributed $r'$.

There are two challenges when adding this noise. First, we must make sure the verifier can still verify such a signature. This can be achieved by giving out $F(r')$ in the signature. Since the output of $F$ is long, we instead give out a short hash $H_1(F(r'))$.

Second, the GCK-OWF is defined over some ring but can only accept inputs that are “short” — i.e., the inputs come from a subset of the ring that are not closed under the homomorphic operation. This makes it challenging to mask the sensitive sum $\sum_j x_{i_j}$. We use the following rejection-sampling approach proposed by Lyubashevsky [145]. Sample the noise $r'$ from a suitable uniform distribution, and restart the entire signing algorithm if the result $\sum_j x_{i_j} + r'$ is “too large” or “too small”. More details about this rejection sampling process are given in §5.3.2.

Finally, instead of choosing indices $i_1, \ldots, i_k$ as $H_2(M)$ as in HORS, we choose them as $H_2(M \| h)$ where $h = H_1(F(r'))$. Intuitively, this ensures that the value $r'$ is “committed” before the
rest of the signature is generated. This aspect of the scheme is used in the security proof, specifically in our use of the generalized forking lemma (Lemma 1). The rewinding argument of the forking lemma implies that any adversary generating a forgery in our scheme can be rewound to output two forgeries with the same \( h \). From these two forgeries, we can break the one-wayness of \( F \).

**Details.** The formal description of the TACHYON scheme is given in Algorithm 6.

\( F_A \) refers to the GCK one-way function discussed in Definition 12. Its input is a vector from \( R^\mu \) and its output is a vector in \( R \), where \( R \) is a suitable ring and \( \mu \) is a small integer. The GCK function is parameterized by a public value \( A \), which is to be chosen randomly. The random choice of \( A \) ensures the one-wayness of \( F_A \) [145, 154]. As such, it may be a global parameter (i.e., shared among all users).

\( \text{Samp}(\gamma) \) samples a uniform distribution over vectors in \( R^\mu \) with all entries in the range \([-\gamma, \gamma]\). This function can easily be implemented with a \text{PRF} or \text{PRG}, similar to other lattice-based constructions that uses uniform sampling (e.g., Dilithium [87]).

\( \text{PRF} \) refers to a pseudorandom function whose output is interpreted as a binary (0/1) vector of \( R^\mu \) (i.e., an input to \( F_A \)).

\( \xi \) and \( \rho \) are parameters related to both the security of the GCK-OWF (controlling the weight of its inputs) as well as the probabilities surrounding rejection sampling (discussed further in §5.3.2).

\( H_1 \) is a random oracle with output length \( l_1 \), used to commit the signature to \( r' \) before choosing the HORS indices. \( H_2 \) is a random oracle with output length \( l_2 = k|t| \) used to choose HORS indices. We write \( \langle i_1 \parallel \cdots \parallel i_k \rangle \leftarrow H_2(M \parallel h) \) to mean that the output of \( H_2 \) is interpreted as a sequence of \( k \) indices, each \( |t| \) bits long.
**Algorithm 6 TACHYON Signature Scheme**

**TACHYON.Kg(1^κ):**

1: \( sk \leftarrow \{0, 1\}^κ \)
2: \( x_i \leftarrow \text{PRF}(sk, i) \), for \( i = 1, \ldots, t \)
3: \( y_i \leftarrow F_A(x_i) \), for \( i = 1, \ldots, t \)
4: return \( sk, pk \leftarrow (t, k, \langle y_1, \ldots, y_t \rangle) \)

**TACHYON.Sig(M, sk):**

1: \( r' \leftarrow \text{Samp}(ξ - 1) \), \( r \leftarrow F_A(r') \)
2: \( h \leftarrow H_1(r) \)
3: \( \langle i_1 \parallel \cdots \parallel i_k \rangle \leftarrow H_2(M \parallel h) \)
4: \( x_{ij} \leftarrow \text{PRF}(sk, i_j) \), for \( j = 1, \ldots, k \)
5: \( s \leftarrow (\sum_{j=1}^{k} x_{ij}) + r' \)
6: if \( \|s\|_\infty \geq (ξ - ρ) \) then goto step 1
7: return \( σ \leftarrow (s, h) \)

**TACHYON.Ver(M, σ, pk):**

1: parse \( σ \) as \( (s, h) \), and \( pk \) as \( (t, k, \langle y_1, \ldots, y_t \rangle) \)
2: if \( \|s\|_\infty \geq (ξ - ρ) \) then return 0
3: \( \langle i_1 \parallel \cdots \parallel i_k \rangle \leftarrow H_2(M \parallel h) \)
4: \( \tilde{r} \leftarrow F_A(s) - \sum_{j=1}^{k} y_{ij} \)
5: if \( H_1(\tilde{r}) = h \) then return 1 else return 0.

**5.3.1.1 Correctness**

TACHYON algorithm is correct in the sense that a signature generated via TACHYON.Sig(·) will always be verified by TACHYON.Ver(·). This can be shown as follows:

Given a message-signature pair \( (M, σ = \langle s, h \rangle) \), due to the deterministic property of the hash oracle \( H_2(·) \) the indexes created in TACHYON.Sig(·) by computing \( \langle i_1 \parallel \cdots \parallel i_k \rangle \leftarrow H_2(M \parallel h) \) are identical to those created in TACHYON.Ver(·). Therefore, given the public key \( pk \leftarrow (t, k, \langle y_1, \ldots, y_t \rangle) \),

\[
F_A(s) - \sum_{j=1}^{k} y_{ij} = F_A((\sum_{j=1}^{k} x_{ij}) + r') - \sum_{j=1}^{k} y_{ij} \\
= F_A((\sum_{j=1}^{k} x_{ij}) + F_A(r') - \sum_{j=1}^{k} F_A(x_{ij}) \\
= F_A(r')
\]
Therefore, for a valid message-signature pair \((M, \sigma = (s, h))\), Step 5 in Algorithm 6 will always return 1.

5.3.2 Rejection Sampling

The idea of rejection sampling in lattices was first proposed by Lyubashevsky [145] in the construction of identification schemes. In our scheme, we need to mask the summation of secret keys \((\sum_j x_{ij})\) with a random \(r'\). If \(r'\) is uniform over the entire ring (on which the summation is defined), then clearly all information about the summation is hidden. However, the verifier must use \(s = \sum_j x_{ij} + r'\) as input to \(F_A\), which is only possible if \(s\) is small. Hence, \(r'\) must be chosen from some bounded distribution. We now discuss how that distribution is determined.

The \(x_i\) vectors are chosen with coefficients from \(\{0, 1\}\). One can easily compute a bound \(\rho\) such that

\[
\Pr \left[ \text{for all subsets } S \text{ with } |S| \leq k: \left\| \sum_{i \in S} x_i \right\|_\infty < \rho \right]
\]

is very high, over the choice of the \(x_i\) values. The rest of the analysis conditions on this highly likely event, and we assume that each coefficient \(a\) of \(\sum_j x_{ij}\) is in the range \(a \in [- (\rho - 1), \rho - 1]\).

Now we choose \(r'\) uniformly with each coefficient in the range \([- (\xi - 1), \xi - 1]\) and set \(s = \sum_j x_{ij} + r'\). This causes each coefficient of \(s\) to be uniform in a range \([a - (\xi - 1), a + \xi - 1]\) for some \(a \in [- (\rho - 1), \rho - 1]\), which depends on the signing key. No matter what \(a\) is, the range \([a - (\xi - 1), a + \xi - 1]\) always contains \([- (\xi - \rho - 1), \xi - \rho - 1]\) as a subrange. Therefore if we condition on all coefficients falling in this subrange, the resulting value is uniform and independent of the signing key. We can achieve this conditioning by rejection sampling, and simply retrying if \(\|s\| \geq \xi - \rho\).
The parameter $\xi$ must be chosen carefully, since larger $\xi$ leads to larger signatures, but smaller $\xi$ leads to more failures/retries during rejection sampling. We can compute the probability of rejection by considering each component of $s$ in isolation. The coefficient is chosen uniformly from some range $[a - (\xi - 1), a + \xi - 1]$, which has $2\xi - 1$ values. The “permissible” outcomes are $[-(\xi - \rho - 1), \xi - \rho - 1]$, a range of $2(\xi - \rho) - 1$ values. Hence the probability that this coefficient is permissible is $\frac{2(\xi - \rho) - 1}{2\xi - 1} = 1 - \frac{2\rho}{2\xi - 1}$. With $\mu N$ coefficients in $s$, the sampling success probability is therefore

$$\left(1 - \frac{2\rho}{2\xi - 1}\right)^{\mu N} \approx e^{-N\mu\rho/\xi}$$

### 5.4 Security Analysis

In the random oracle model [34], we prove that TACHYON is $EU$-CMA in Theorem 4 below. Note that in our proof, we ignore terms that are negligible in terms of our security parameter.

**Theorem 4.** In the random oracle model, if there exists an adversary $A$ that can $(t_A, q_S, q_H, \epsilon_A)$-break the $EU$-CMA security of TACHYON, then one can build another algorithm $C$, that can break the one-wayness of the GCK function family (as defined in Definition 12) with success probability of at least

$$\frac{1}{t} \left[ \left( \epsilon_A - \frac{q_H(q_S + q_H)}{2^{l_1}} \right) \left( \frac{\epsilon_A}{q_H} - \frac{q_S + q_H}{2^{l_1}} - \frac{1}{2^{l_2}} \right) - \frac{q_H^{k!}}{2^{l_2}} \right]$$

and running in time at most

$$O(2t_A + t(t_{RNG} + t_{F_A}) + q_S(2t_{RNG} + t_{F_A} + k_{Add}) + q_H t_{RNG})$$

where $t_{RNG}$, $t_{Add}$ and $t_{F_A}$ are the running time of a random number generator, vector addition and $F_A$ function, respectively.
The intuition behind the reduction is as follows. The reduction algorithm receives a value \( y^* \) and attempts to find a preimage of \( y^* \) under \( F_A \). The reduction algorithm plays the role of the challenger (EU-CMA game) against \( A \), and uses \( y^* \) as one of the public-key components \( y^*_j \), for random index \( j^* \). It chooses all other public-key components \( y_i \) honestly.

The reduction algorithm does not know the entire signing key (it does not know \( x^*_j \)), so it uses its ability to program the random oracle to generate simulated signatures. Specifically, it chooses the signature \((s, h)\) uniformly at random, and then programs \( H_1 \) and \( H_2 \) so that the signature verifies.

Suppose \( A \) successfully constructs a forgery \((s, h)\). Consider rewinding the adversary to the point where it made the query \( H_2(M \parallel h) \), then continuing with independent randomness. The forking lemma states that, with good probability, the adversary will output a forgery \((s', h)\) in this case as well. Importantly, the new forgery will include the same \( h \), hence:

\[
h = H_1 \left( F_A(s) - \sum_{j \in I} y_j \right) = H_1 \left( F_A(s') - \sum_{j \in I'} y_j \right)
\]

Note that the two summations are over different multisets \( I, I' \) of indices.

Conditioning on the absence of a collision in \( H_1 \), we have

\[
F_A(s) - \sum_{j \in I} y_j = F_A(s') - \sum_{j \in I'} y_j
\]

Say that \( I \) and \( I' \) are compatible if there is some index that appears with multiplicity exactly once in \( I \cup I' \). Our reduction conditions on the fact that \( I \) and \( I' \) are always compatible. With independent probability \( 1/t \), we have that \( I \) and \( I' \) are actually compatible with respect to our special index \( j^* \).
Compatibility implies that we can solve for $y^\ast$. Let's say $j^\ast \in I \setminus I'$, then:

$$y^\ast = F_A(s') - F_A(s) + \sum_{j \in I \setminus \{j^\ast\}} y_j - \sum_{j \in I'} y_j$$

The reduction algorithm knows the preimages to all $y_j$ terms on the right-hand side. It is therefore possible to apply the homomorphic property of $F_A$ and write the right-hand side as $F_A$ applied to a value known to the reduction algorithm. In other words, the reduction can compute a preimage of $y^\ast$.

5.4.1 Compatible index sets

Before describing the reduction in more detail, we clarify the properties of compatible index sets.

*Definition 20.* Let $I, I'$ be strings which encode multisets in the natural way as $I = \langle i_1 | \cdots | i_k \rangle$, etc. We say that $I$ and $I'$ are *compatible with respect to* $i$ if $i$ appears with multiplicity 1 in $I$ and multiplicity 0 in $I'$ (or vice-versa). We say that $I$ and $I'$ are *compatible* if they are compatible for some value $i$.

Each $I$ encodes $k$ indices. In the worst case there are at most $k!$ other strings that encode a multiset that is *incompatible* with $I$. If we have one fixed string $I^\ast$ and $q$ other uniformly chosen strings $I_1, \ldots, I_q$ (all strings with $l_2$ bits)

$$\Pr[I^\ast \text{ is compatible with all } I_1, \ldots, I_q] \geq \left(1 - \frac{k!}{2^{l_2}}\right)^q \geq 1 - \frac{q \cdot k!}{2^{l_2}}$$
and hence:

\[ \Pr[I^* \text{ is not compatible with all } I_1, \ldots, I_q] \leq \frac{q \cdot k!}{2^{l_2}} \]

We abbreviate the latter probability as \( \Pr[\text{Compat}(q, k, l_2)] \).

**Reduction algorithm.** Given an adversary \( A \), we define the reduction algorithm/game \( B \) in Algorithm 7. \( B \) takes \( y^* \) (an \( F_A \)-output) as input, as well as a list \( H \) of random oracle responses that it will use to program \( H_2 \). This interface is necessary for our usage of the forking lemma.

\( B \) proceeds to simulate the EU-CMA game against \( A \), implanting \( y^* \) within the public key and generating simulated signatures as described above.

If \( A \) is successful in generating a forgery, then \( B \) outputs it, as well as the index of the hash call corresponding to \( H_2(M^* || h^*) \). This indicates to the forking lemma that we wish to rewind to this query and resume with fresh randomness.

**Claim 1.** \( \Pr[\text{FORGERY}] \geq \epsilon_A - \frac{q \cdot k^2 + q^2}{2^{l_1}} + \text{negl}(\kappa) \), where the negligible quantity is from the security of PRF.

**Proof.** First, we compare the view of \( A \) in the reduction to its view in the standard EU-CMA game. The only differences are:

1. The \( x_i \) values are chosen uniformly rather than pseudorandomly. This changes the adversary’s view by a negligible amount.

2. The signature is generated in “reverse order”. From the discussion in §5.3.2, real signatures are distributed uniformly, hence this difference has no effect on the adversary’s view.

Overall, we see that the adversary’s view is indistinguishable.
Algorithm 7 Reduction algorithm $B$.

1: function $\text{INITIALIZE}(y^*, H)$
2: \hspace{1em} $j^* \xleftarrow{\$} \{1, \ldots, t\}$
3: \hspace{1em} $y_{j^*} \leftarrow y^*$
4: \hspace{1em} $x_i \xleftarrow{\$} \text{Samp}(1)$, for $i \in \{1, \ldots, t\} \setminus \{j^*\}$
5: \hspace{1em} $y_i \leftarrow F_A(x_i)$, for $i \in \{1, \ldots, t\} \setminus \{j^*\}$
6: \hspace{1em} return $pk \leftarrow (t, k, \langle y_1, \ldots, y_t \rangle)$
7: end function
8: function $H_1(q)$
9: \hspace{1em} if $L_1[q]$ is not defined then
10: \hspace{2em} $L_1[q] \xleftarrow{\$} \{0, 1\}^{l_1}$
11: \hspace{1em} return $L_1[q]$
12: end function
13: function $H_2(q)$
14: \hspace{1em} if $L_2[q]$ is not defined then
15: \hspace{2em} $L_2[q] \leftarrow$ next unused value from $H$
16: \hspace{1em} return $L_2[q]$
17: end function
18: function $\text{SIG}(M)$
19: \hspace{1em} add $M$ to set $M$
20: \hspace{1em} $s \xleftarrow{\$} \text{Samp}(\xi - \rho - 1)$
21: \hspace{1em} $h \xleftarrow{\$} \{0, 1\}^{l_1}$
22: \hspace{1em} $I = \langle i_1 \ldots i_k \rangle \leftarrow$ next unused value from $H$
23: \hspace{1em} $\tilde{r} \leftarrow F_A(s) - \sum_{j=1}^{k} y_{i_j}$
24: \hspace{1em} if $L_1[\tilde{r}]$ or $L_2[M||h]$ are defined then $\text{BAD1} \leftarrow 1$; abort
25: \hspace{1em} $L_1[\tilde{r}] \leftarrow h$
26: \hspace{1em} $L_2[M||h] \leftarrow I$
27: \hspace{1em} return $(s, h)$
28: end function
29: function $\text{FINALIZE}(M^*, \sigma^* = (s^*, h^*))$
30: \hspace{1em} if there is a duplicate value in $L_1$ then $\text{BAD2} \leftarrow 1$; abort
31: \hspace{1em} if $[M^* \not\in M] \land [\text{SGN.Ver}(M^*, \sigma^*, pk) = 1]$ then
32: \hspace{2em} FORGERY $\leftarrow 1$
33: \hspace{2em} let $v$ be the index such that $L_2[M^*||h^*] = H[v]$
34: \hspace{1em} return $(v, \sigma^*)$
35: else
36: \hspace{1em} return $(0, 0)$
37: end function

The only other difference between the reduction and EU-CMA game is that the reduction may abort in the event of $\text{BAD1}$ or $\text{BAD2}$. $\text{BAD1}$ happens when the reduction needs to program the random oracles but they have already been queried on the desired point. On line 21, the values $\tilde{r}$
and $h$ are uniform, each with at least $l_1$ bits of entropy. Hence the probability that such a prior query has been made is at most $q_H/2^{l_1}$. Taking a union bound over all $q_S$ calls to $S1G$, the overall probability of $BAD1$ is bounded by $q_Sq_H/2^{l_1}$.

$BAD2$ happens when a collision is found in $H_1$. This probability is bounded by $q^2/2^{l_1}$.

Forking lemma. Now, we can consider invoking the forking lemma (Lemma 1) with $B^A$. The result is an algorithm $Fork_B$ that has probability at least

$$\Pr[\text{FORGERY}] \left( \frac{\Pr[\text{FORGERY}]}{q_H} - \frac{1}{2^{l_2}} \right)$$

of producing two forgeries. Note that these forgeries must be with respect to the same $M^*$ and $h^*$ values because of the way that $B$ computes the index $v$ of the “special” oracle query, and the fact that the forking lemma ensures that this index is the same in both “forks.” Each forgery verifies with respect to a different value of $H_2(M^*\|h^*)$.

Claim 2. Let $\sigma_1^* = (s_1^*, h^*)$ and $\sigma_2^* = (s_2^*, h^*)$ be the two forgeries output by $Fork_B$, for message $M^*$. Let $I_1$ be the value of $H_2(M^*\|h^*)$ in the first “fork” and $I_2$ be its value in the second “fork.” When $I_1$ and $I_2$ are compatible with respect to $j^*$, a preimage of $y^*$ can be computed efficiently.

Proof. Following the high-level discussion, we can solve for a preimage of $y^*$. Write $I_1 = \langle i_1^{(1)} \| \cdots \| i_k^{(1)} \rangle$ and $I_2 = \langle i_1^{(2)} \| \cdots \| i_k^{(2)} \rangle$.

By symmetry, suppose $j^*$ appears in $I_1$ but not $I_2$. From the verification equation for these signatures we have:

$$h^* = H_1 \left( F_A(s_1^*) - \sum_{j=1}^k y_{i_j^{(1)}} \right) = H_1 \left( F_A(s_2^*) - \sum_{j=1}^k y_{i_j^{(2)}} \right)$$
Since $\mathcal{B}$ aborts if a collision was found in $H_1$ (BAD2 event), we have

$$F_A(s_1^*) - \sum_{j=1}^{k} y_{i_j(1)} = F_A(s_2^*) - \sum_{j=1}^{k} y_{i_j(2)}$$

Isolating $y_{j^*} = y^*$ (which appears in the left summation but not the right one) and using the homomorphic property of $F_A$ gives:

$$y^* = F_A(s_1^*) - F_A(s_2^*) - \sum_{j=1}^{k} y_{i_j(1)} + \sum_{j=1}^{k} y_{i_j(2)}$$

$$= F_A(s_1^*) - F_A(s_2^*) - \sum_{j=1}^{k} F_A(x_{i_j(1)}) + \sum_{j=1}^{k} F_A(x_{i_j(2)})$$

$$= F_A \left( s_1^* - s_2^* - \sum_{j=1}^{k} x_{i_j(1)} + \sum_{j=1}^{k} x_{i_j(2)} \right)$$

The final argument to $F_A$ is a value that can be computed from known values, and it is a preimage of $y^*$.

Proof of Theorem 4. Given an adversary $\mathcal{A}$ breaking EU-CMA security as stated, we first construct the reduction algorithm/game $\mathcal{B}$ (Algorithm 7). From Claim 1, the game produces a forgery with probability (ignoring negligible terms related to the PRF):

$$\Pr[\text{FORGERY}] \geq \epsilon_{\mathcal{A}} - \frac{q_H q_S + q_H^2}{2^{l_1}}$$
We then apply the forking lemma (Lemma 1) to $B^A$. The result is an algorithm $\text{Fork}_B$ that generates two forgeries with probability at least:

$$\Pr[\text{FORKERY}] \left( \frac{\Pr[\text{FORKERY}]}{q_H} - \frac{1}{2^l_2} \right)$$

In the event that $\text{Fork}_B$ outputs two forgeries, define $I_1$ to be the value of $H_2(M^*||h^*)$ in the first “fork” and $I_2$ to be its value in the second “fork.” Looking ahead, we would like to bound the probability that $I_1$ and $I_2$ are compatible. However, we run into a problem because $I_2$ is not distributed independently of $\text{Fork}_B$’s success. Intuitively, the adversary gets to “choose” whether the second fork succeeds after seeing $I_2$.

On the other hand, let $\mathcal{H}'$ be the set of oracle responses that are re-sampled uniformly during the second “fork.” Importantly, $\mathcal{H}'$ is distributed independently of $I_1$, so we can bound the probability that $I_1$ is compatible with all elements of $\mathcal{H}'$. Since $I_2$ (if it exists) is guaranteed to be an element of $\mathcal{H}'$, this allows us to reason about the compatibility of $I_1$ and $I_2$.

From these observations, we obtain:

$$\Pr[\text{preimage of } y^* \text{ is found}]$$

$$= \Pr[\text{Fork}_B \text{ outputs 2 forgeries and } I_1, I_2 \text{ are compatible wrt } j^*]$$

$$= \frac{1}{t} \Pr[\text{Fork}_B \text{ outputs 2 forgeries and } I_1, I_2 \text{ are compatible}]$$

$$\geq \frac{1}{t} \Pr[\text{Fork}_B \text{ outputs 2 forgeries and } I_1, \mathcal{H}' \text{ are compatible}]$$

$$\geq \frac{1}{t} \left( \Pr[\text{Fork}_B \text{ outputs 2 forgeries}] - \Pr[\text{Compat}(q_H,k,I_2)] \right)$$

$$\geq \frac{1}{t} \left[ \Pr[\text{FORKERY}] \left( \frac{\Pr[\text{FORKERY}]}{q_H} - \frac{1}{2^l_2} \right) - \frac{q_H k!}{2^l_2} \right]$$

$$= \frac{1}{t} \left[ \left( \epsilon_A - \frac{q_H (q_S + q_H)}{2^l_1} \right) \left( \frac{\epsilon_A}{q_H} - \frac{q_S + q_H}{2^l_1} - \frac{1}{2^l_2} \right) - \frac{q_H k!}{2^l_2} \right]$$

72
Note that the third line follows from the fact that the adversary's view in $B^A$ is independent of $j^*$. The running time of $C$ is that of $A$ to output two forgery signatures with an overwhelming probability plus the time it takes for the simulation processes. For the sake of convenience, we do not consider the negligible processes. The setup process takes $t \cdot (t_{RNG} + t_{FA})$, where $t$ is the HORS parameter, for generating private keys and the corresponding public keys. Each signing process would require $2t_{RNG}$ to generate $r'$ and $I = (i_1, \ldots, i_k)$ one $t_{FA}$ and $k \cdot t_{Add}$. Each hash query would require a $t_{RNG}$. Therefore, the total running time of $C$ is upper-bounded by

$$O(2t_A + t(t_{RNG} + t_{FA}) + q_S(2t_{RNG} + t_{FA} + kt_{Add}) + q_H t_{RNG})$$

This completes the proof.

5.4.2 Parameters

In this section, we discuss parameter choices for our construction as shown in Table 5.1.

5.4.2.1 Collision-Freeness of GCK Function

For TACHYON, $N$ and $\mu$ are 256 and 8, respectively. As it has been shown in [147, 170], for the family of GCK functions to admit a strong security reduction, one needs to ensure that $\mu > \frac{|q|}{2^d}, q > 4d\mu N^{1.5}|N|$ for domain $D = \{g \in R : \|g\|_\infty \leq d\}$ for some value $d$. Specifically, based on the analysis in [147, 155, 170], with these parameters, finding collision on average (when $a_{i,j} \in \mathbb{Z}_q$) with any non-negligible probability is at least as hard as solving the underlying problem (i.e., $SPP_\gamma(I)$ [147]) on certain kinds of point lattices, in the worst-case. We note that our concrete parameter selection, as provided in the following, meets the requirements stated above to allow for a strong security reduction.
5.4.2.2 Lattice Attacks

Given a uniformly random vector \( \mathbf{a} = (a_1, \ldots, a_\mu) \in \mathbb{R}^\mu \), the SIS problem over a ring asks to find a non-zero vector \( \mathbf{x} = (x_1, \ldots, x_\mu) \in \mathbb{Z}[x]/(x^N+1) \) such that

\[
\sum_{i=1}^m a_i x_i \equiv 0 \mod q, \text{ where } \|\mathbf{x}\| \leq \beta
\]

An approach to estimate the hardness of this problem is by measuring the run-time of lattice basis reduction algorithms. These reduction algorithms aim to find the nice bases which consist of reasonably short and (nearly) orthogonal vectors. Gama and Nguyen [100] show that such reduction algorithms for a lattice \( \mathcal{L} \) with dimension \( N \) can find vectors of length \( \leq \delta^N \cdot \det(\mathcal{L})^{\frac{1}{N}} \) where \( \delta \) is the Hermite delta. The BKZ algorithm [183] is the best known algorithm for finding short (non-zero) vectors in lattices. The BKZ algorithm starts by reducing a lattice basis using a Shortest Vector Problem (SVP) oracle in a smaller dimension. As shown in [115], the number of calls to the SVP oracle remains polynomial, however, precisely computing the number of calls is an arduous task and therefore, subject to heuristic approaches (e.g., BKZ 2.0 [73]). BKZ 2.0 requires solving the SVP problem in lattices with dimension at most \( b < N \), where \( b \) is called the block size. Therefore, BKZ 2.0 runs for multiple rounds to find the final output. Given the norm bound \( \beta \) of an SIS instance, the corresponding \( \delta \) can be computed as

\[
\delta = \left( \frac{b \cdot (\pi b)^{\frac{1}{2}}}{2\pi e} \right)^{\frac{1}{2(N-b)}}
\]

The most recent classical solver for SVP [25] runs in time \( \approx 2^{0.292b} \) and the best known quantum solver for SVP [135] runs in time \( \approx 2^{0.265b} \).
In the following we discuss our estimation based on the works in [12, 13, 87, 111].

We consider two types of adversary powers, namely, the classical and post-quantum. For TACHYON, we proffer three parameter sets (for three security levels) and analyze the security level of each for the adversarial types mentioned above. In the classical model, for our medium security instantiation, we set $q = 2^{27} - 2^{11} + 1$ and $\beta = 2^{16}$ to achieve $\delta \approx 1.00339$ with $b = 502$. We set $q = 2^{30} - 2^{18} + 1$ and $\beta = 2^{17}$ for recommended instantiation which achieves $\delta \approx 1.00271$ with $b = 682$. We set $q = 2^{31} - 2^{9} + 1$ and $\beta = 2^{17}$ for the high security instantiation with $\delta \approx 1.00203$ with $b = 1007$. Therefore, based on the analysis in [13, 87], we achieve 146, 199 and 294 classical bit security for the medium, recommended and high security instantiations of TACHYON against lattice attacks, respectively. For post-quantum security against lattice attacks, we achieve 133, 180 and 266 bit security for the medium, recommended and high security instantiations, respectively. Similar to Dilithium [87], our parameter choices are conservative.

5.4.2.3 $k$-Element Combinatorial Problem

As captured in our security proof, $k, t$ parameters must be selected such that the probability $\frac{2^{k|t|} k!}{2^2}$ is negligible. Considering that $l_2 = k|t|$ (since $k$ indexes that are $|t|$-bit long are selected with the hash output), this gives us $\frac{2^{k|t|} k!}{2^2|t|}$. We further elaborate on some choices of $(k, t)$ along with their security/performance implications in §5.5.

5.4.2.4 Quantum Random Oracle Model (QROM)

QROM considers the scenario where the adversary has classical access to the signing oracle and quantum access to the hash function oracle. TACHYON is proven to be secure in the random oracle model and we do not provide the proof for the security of TACHYON in QROM. This trend is true for a wide range of "efficient" schemes (e.g., [87]), which are mostly based on Fiat-Shamir framework, since
Table 5.1: Parameter selection of TACHYON

<table>
<thead>
<tr>
<th>Parameter</th>
<th>TACHYON-128</th>
<th>TACHYON-192</th>
<th>TACHYON-256</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N$</td>
<td>256</td>
<td>256</td>
<td>256</td>
</tr>
<tr>
<td>$\mu$</td>
<td>8</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>$q$</td>
<td>134215681</td>
<td>1073479681</td>
<td>2147483137</td>
</tr>
<tr>
<td>$t$</td>
<td>1024</td>
<td>2048</td>
<td>3072</td>
</tr>
<tr>
<td>$k$</td>
<td>18</td>
<td>25</td>
<td>32</td>
</tr>
<tr>
<td>$l_1$</td>
<td>256 bits</td>
<td>384 bits</td>
<td>512 bits</td>
</tr>
<tr>
<td>$l_2$</td>
<td>180 bits</td>
<td>275 bits</td>
<td>384 bits</td>
</tr>
<tr>
<td>RS Rate(^\d)</td>
<td>3.08</td>
<td>2.18</td>
<td>2.72</td>
</tr>
</tbody>
</table>

\(^\d\) RS Rate denotes Rejection Sampling Rate.

their ROM is not "history free" due to the forking lemma in the reduction step. Initial approaches (e.g., [200]) to obtain QROM security for schemes based on Fiat-Shamir transformation resulted in considerably less efficient signatures since they needed multiple execution of the underlying identification scheme. However, recently, in line of providing QROM security for Dilithium [87], Kiltz et al. [130] provide a tight reduction in the QROM which incurs less performance/storage penalty as compared to directly applying the method in [200]. This generic framework [130] can be applied to the identification schemes that admit lossy public keys. We believe it is possible to prove the security of TACHYON in QROM and therefore, in the line of Dilithium [87] and its QROM secure instantiation [130], we will investigate the QROM security of TACHYON in our future work.

5.5 Performance Evaluation

We first present analytical performance analysis and some of the potential performance/speed trade-offs for TACHYON. We then provide our evaluation metrics and experimental setup followed by a detailed experimental comparison of TACHYON with the state-of-the-art PQ-secure digital signature schemes.
5.5.1 Analytical Performance Analysis

We now describe the analytical performance of our scheme based on the parameters. In the computational overhead analysis, we present our runtime in terms of the total number of PRF, GCK function, and vector addition calls. We omit the overhead of small-constant number of hash calls.

- **Signer Computation and Storage Overhead:** TACHYON only requires storing a $\kappa$-bit random seed number as the private key, which is used to deterministically generate the required $x_i$ components via PRF calls, where each $x_i$ is $\mu \cdot N$ bits. The signature generation cost is significantly affected by the derivation and summation of $k$ number of $x_i$. This requires $k \cdot \text{PRF}$ calls, extracting the binary vectors from the PRF outputs and vector additions (whose computational overhead is negligible). For each PRF call, a $\kappa$-bit input is extended to a $\mu \cdot N$ bit output. In addition, a $\text{Samp}(\xi - 1)$ function is required. $\text{Samp}(\xi - 1)$ generates a vector of length $\mu \cdot N$ with components of length $|\xi|$ bits. Therefore, $\text{Samp}(\xi - 1)$ can be implemented with a PRF that extends a $\kappa$-bit input to a $|\xi| \cdot \mu \cdot N$ bit output. In total, these correspond to the generation of $(|\xi| + k) \cdot \mu \cdot N$ pseudorandom bits via a PRF. Another significant cost for signature generation is the GCK function call that is made to compute the image of the randomness $r'$. A GCK call is basically composed of two operations: Number Theoretic Transform (NTT) calculation and a linear combination. In order to compute a GCK call, $\mu$ number of NTT calls and a single linear combination is necessary, where both of these operations are based on simple multiplications and additions under $\text{mod} \ q$. Therefore, in total, TACHYON signature generation requires storing $\kappa$-bit of private key, $k$ PRF invocations, $k$ vector additions, a single $\text{Samp}(\xi - 1)$ and a GCK function call to compute a signature.

- **Signature Size:** The signature $\sigma$ is comprised of the vector $s$ and a hash output $h$, where $|h| = l_1$. Rejection sampling enforces $s$ to satisfy $\|s\|_\infty < \xi - \rho$. Since $s$ consists of $\mu \cdot N$
components, this vector can be represented with $|\xi - \rho| \cdot \mu \cdot N$ bits. The total size of a signature is $|\xi - \rho| \cdot \mu \cdot N + l_1$ bits.

- **Verifier Computation and Storage Overhead:** The signature verification requires only a single GCK call and $k$ vector additions, which makes it the most verifier computationally efficient scheme among its current counterparts. On the other hand, the size of public key is $|q| \cdot \mu \cdot N \cdot t$ bits (i.e., $t$ vectors of length $\mu \cdot N$), which is relatively larger than its counterparts.

- **Improved Side-Channel Resiliency:** TACHYON only requires a uniform sampling $\text{Samp}(\xi - 1)$ in its signature generation. Since it does not require Gaussian sampling, it has an improved side-channel resiliency as compared to some of its lattice-based counterparts (e.g., BLISS [85]). Moreover, the rejection sampling in BLISS is based on iterated Bernoulli trials, that is prone to some attacks. As it is shown in [94], this efficient rejection sampling technique has been exposed to some side channel attacks. Although, TACHYON requires rejection sampling to make sure the statistical distribution of the signatures does not leak information about the private key components, similar to [87], since our rejection sampling does not require any Bernoulli trials, the attack does not apply to our rejection sampling step.

5.5.2 Performance-vs-Storage Trade-offs

Our design allows several trade-offs between performance and storage that may be suitable for different use-cases.

- **Signer Pre-computation:** With a basic implementation trick, one can store the $x_i$’s instead of deterministically generating them at the signature generation. This enables the signer to avoid the cost of generating these values ($k \cdot \text{PRF}$ calls, and extracting the binary vectors) during the signature generation. Since the signer must store these $x_i$ vectors, this adds up to a private
key of at least $t \cdot \mu \cdot N$ bits, that is larger than that of TACHYON. However, this caching strategy offers a faster signature generation and therefore can be preferred when the signer is able to store such vectors. Signature generation speed advantages and required private key size are further explained in §5.5.4. Specifically, the signer must store three types of vectors with the size of bit,

- **Selection of $t, k$:** The parameter $t$ linearly impacts the size of public key of TACHYON. The parameter $k$ determines the number of PRF calls, binary vectors to be extracted and vector additions in TACHYON signing, and also the number of vector additions in TACHYON signature verification. Note that decreasing $t$ requires an increase in $k$ (or vice versa) to preserve the desired security level. We selected $(t = 1024, k = 18)$, $(t = 2048, k = 25)$, and $(t = 3072, k = 32)$ to provide $\kappa = 128$-bit, $\kappa = 192$-bit, and $\kappa = 256$-bit security, respectively. However, different parameters for the same security levels are also possible. For instance $t = 256, k = 26$ would also offer $\kappa = 128$-bit security level and could be preferred (over $t = 1024, k = 18$) for TACHYON medium level security instantiation. This would provide a $4 \times$ smaller public key, where the signature generation time would be increased.

- **Rejection Sampling Parameters:** Rejection sampling rate implies how many times (on average) the signature generation should be executed to output an “acceptable” signature. Therefore, the increment of the acceptance probability has a linear effect on the signature generation time. We discuss two parameters that can be tuned to increase the acceptance probability of the outputted signatures, (i) increasing $\xi - 1$ (where $\xi - 1 = \|r'\|_\infty$), and (ii) decreasing $k$. While tuning these parameters can result in significantly decreasing the average signing time, there are trade-offs to consider. Increasing $\xi - 1$ causes an increase on the signature size. Additionally, this increase incurs a security loss as it directly affects the hardness of the
Table 5.2: Experimental performance comparison of **TACHYON** with its counterparts

<table>
<thead>
<tr>
<th>Scheme</th>
<th>Security Level (bit)</th>
<th>Signature Gen Time (µs)</th>
<th>Private Key (Byte)</th>
<th>Signature Size (Byte)</th>
<th>Signature Ver Time (µs)</th>
<th>Public Key (Byte)</th>
<th>End-to-End Delay (µs)</th>
<th>Gaussian Sampling</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPHINCS [123]</td>
<td>128</td>
<td>14265</td>
<td>64</td>
<td>16976</td>
<td>617</td>
<td>32</td>
<td>13242</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td>192</td>
<td>18580</td>
<td>96</td>
<td>35664</td>
<td>974</td>
<td>48</td>
<td>19534</td>
<td></td>
</tr>
<tr>
<td></td>
<td>256</td>
<td>42898</td>
<td>128</td>
<td>49216</td>
<td>1015</td>
<td>64</td>
<td>43913</td>
<td></td>
</tr>
<tr>
<td>pqsigRM [139]</td>
<td>128</td>
<td>3960</td>
<td>128218</td>
<td>260</td>
<td>21</td>
<td>33804</td>
<td>3981</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td>192</td>
<td>29260</td>
<td>334006</td>
<td>516</td>
<td>30</td>
<td>501176</td>
<td>26290</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td>256</td>
<td>406</td>
<td>2105344</td>
<td>1028</td>
<td>138</td>
<td>2144166</td>
<td>544</td>
<td></td>
</tr>
<tr>
<td>GeMSS [69]</td>
<td>128</td>
<td>252844</td>
<td>14208</td>
<td>48</td>
<td>39</td>
<td>417408</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td></td>
<td>192</td>
<td>776330</td>
<td>39440</td>
<td>199</td>
<td>3603792</td>
<td>118868</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>256</td>
<td>406</td>
<td>2105344</td>
<td>1028</td>
<td>138</td>
<td>2144166</td>
<td>544</td>
<td></td>
</tr>
<tr>
<td>PICNIC [71]</td>
<td>128</td>
<td>1966</td>
<td>16</td>
<td>3400</td>
<td>1355</td>
<td>32</td>
<td>3301</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td>192</td>
<td>6951</td>
<td>24</td>
<td>76740</td>
<td>4804</td>
<td>48</td>
<td>11755</td>
<td></td>
</tr>
<tr>
<td></td>
<td>256</td>
<td>1393</td>
<td>32</td>
<td>132824</td>
<td>9639</td>
<td>64</td>
<td>23602</td>
<td></td>
</tr>
<tr>
<td>Bliss [85]</td>
<td>128</td>
<td>141</td>
<td>256</td>
<td>717</td>
<td>28</td>
<td>896</td>
<td>163</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>192 ✴</td>
<td>211</td>
<td>384</td>
<td>768</td>
<td>28</td>
<td>896</td>
<td>239</td>
<td></td>
</tr>
<tr>
<td></td>
<td>256 ✴</td>
<td>392</td>
<td>384</td>
<td>813</td>
<td>31</td>
<td>896</td>
<td>423</td>
<td></td>
</tr>
<tr>
<td>qTESLA [48]</td>
<td>128</td>
<td>650</td>
<td>1856</td>
<td>2720</td>
<td>133</td>
<td>2976</td>
<td>783</td>
<td></td>
</tr>
<tr>
<td></td>
<td>192</td>
<td>2524</td>
<td>4160</td>
<td>5664</td>
<td>272</td>
<td>6176</td>
<td>2796</td>
<td></td>
</tr>
<tr>
<td></td>
<td>256</td>
<td>6703</td>
<td>4128</td>
<td>5920</td>
<td>314</td>
<td>6432</td>
<td>7127</td>
<td></td>
</tr>
<tr>
<td>pqNTRUsign [120]</td>
<td>128</td>
<td>14516</td>
<td>1024</td>
<td>576</td>
<td>1024</td>
<td>1024</td>
<td>14820</td>
<td>×</td>
</tr>
<tr>
<td></td>
<td>192</td>
<td>2156</td>
<td>2800</td>
<td>2044</td>
<td>53</td>
<td>1184</td>
<td>219</td>
<td></td>
</tr>
<tr>
<td></td>
<td>256</td>
<td>272</td>
<td>3504</td>
<td>2701</td>
<td>76</td>
<td>1472</td>
<td>348</td>
<td></td>
</tr>
<tr>
<td>Dilithium [87]</td>
<td>128</td>
<td>138</td>
<td>16</td>
<td>4416</td>
<td>18</td>
<td>884136</td>
<td>156</td>
<td></td>
</tr>
<tr>
<td></td>
<td>192</td>
<td>124</td>
<td>24</td>
<td>4672</td>
<td>21</td>
<td>1966080</td>
<td>145</td>
<td></td>
</tr>
<tr>
<td></td>
<td>256</td>
<td>198</td>
<td>32</td>
<td>4672</td>
<td>28</td>
<td>3047424</td>
<td>226</td>
<td></td>
</tr>
</tbody>
</table>

† **TACHYON** requires rejection sampling in its signature generation (similar to **BLISS** [85], **Dilithium** [87]). The number of required signature generation repetitions due to rejection sampling are 3.08, 2.18 and 2.72 for medium, recommended and high security levels, respectively.

‡ Gaussian sampling requirement is same for the *all security levels*, and therefore, it is represented with a single value.

✓ Denotes the scheme requires Gaussian sampling, that can be considered unfavorable due to the side-channel attacks.

* Denotes security level other than standard 128, 192, 256 bits.

lattice attacks discussed in §5.4.2.2. On the other hand, as discussed above, decreasing $k$ would require increasing $t$ to compensate for the security loss, that increases the public key size.

### 5.5.3 Experimental Evaluation and Setup

We describe our experimental evaluation metrics and setup, wherein our scheme and their counterparts are compared with each other.

- **Evaluation Metrics**: We have evaluated and compared **TACHYON** with its counterparts in terms of signature generation and verification times, private key, public key and signature sizes and end-to-end cryptographic delay (i.e., the sum of signature generation and verification times, excluding the signature transmission time, as it is network depended).
• **Hardware Configurations**: We used a laptop equipped with an Intel i7 6th generation (Skylake) 2.6GHz processor and 12 GB of RAM for our experiments.

• **Implementation Details**: Our parameter selection which is based on [147] - i.e., $N$ is a power-of-two and $1 \equiv q \mod (2N)$ - allows us to use NTT to accelerate the GCK function computations. Similar approach has been done in [87]. Then, to finalize the GCK function, we computed the linear combination under $\mod q$ of input with random and public matrix $A$. Since highest $|q|$ selected is just 31, we did not use any libraries for these calculations. We would like to note that this operation can be performed very fast with some assembly level optimizations. However, in this work, we used a conservative implementation.

We instantiated $H_1$ and $H_2$ random oracles using BLAKE2b due to its optimization for commodity hardware, in terms of speed and security [21]. We used Intel intrinsics to implement our PRF function and $\text{Samp}(\xi - 1)$ (with AES in counter mode). Our implementation is open-sourced in the following link.

https://github.com/ozgurozmen/TACHYON

For our counterparts, we used the optimized codes (if available, otherwise the reference codes) that are submitted to the NIST competition and ran them on our processor. Note that among all the schemes presented in Table 5.2, only BLISS is not a NIST competitor. For this scheme, we used the open-sourced implementation provided by the authors.

5.5.4 Performance Analysis and Comparison

Table 5.2 shows the experimental performances of TACHYON and its state-of-the-art counterparts. We selected various schemes that are submitted to the first NIST post-quantum cryptography standardization conference (except BLISS [85], that is selected since it is one of the
fastest lattice-based signatures). These schemes include lattice-based constructions (qTESLA [48], pqNTRUsign [120], and Dilithium [87]), a hash-based construction (SPHINCS+ [123]), a code-based construction (pqsigRM [139]), a symmetric key cryptography based construction (PICNIC [71]) and a multivariate-based scheme (GeMSS [69]).

Table 5.2 shows that TACHYON has the lowest end-to-end delay and both its signature generation and verification are the fastest among its counterparts, for every security level. For instance TACHYON-192 has the fastest signature generation and the lowest end-to-end delay among all the schemes with any security level. Moreover, TACHYON offers the lowest possible private key size (that is the same with symmetric key based PICNIC). TACHYON has a signature of slightly more than 4 KB, that is comparable to its lattice-based counterparts but larger than multivariate and code-based constructions. TACHYON public key is significantly larger than most of their counterparts (only smaller than GeMSS in high security levels). Considering the overall efficiency of TACHYON, we believe it can be preferred when the verifier can tolerate such a storage.

As discussed in §5.5.2, one can consider caching the $x_i$ vectors as the private key instead of deterministically deriving them with a $\kappa$-bit seed. When this optimization is considered, it provides a signature generation that is significantly faster than that of TACHYON. With the verification being unchanged, this variant can further improve the end-to-end delay (which is currently the fastest). On the other hand, when these vectors are cached, the private key size increases significantly (e.g., $256 - 768$ KB), that is only smaller than pqsigRM, for certain security levels. This can make caching impractical for some applications where the signer is memory-limited. In these cases, TACHYON without any caching should be preferred.

We also dissected the cost of TACHYON, for future optimizations. GCK function computation corresponds to the $\approx 40\%$ of the total cost for TACHYON-128 signature generation, that slightly
decreases on higher security levels. The highest cost is identified as the PRF calls and the extraction of the binary vectors from this PRF output, made to deterministically generate the vectors ($x_i$’s). This can be further confirmed with the improvements observed by caching the $x_i$ vectors, where this cost is eliminated and replaced with only vector additions. For the signature verification, over 80% of the total cost is due to the GCK function.

5.5.4.1 Discussions

The GCK function calculations can be further accelerated with assembly instructions on NTT function as in Dilithium [87]. In this chapter, we presented our benchmark results with a reference implementation, without any assembly level instructions. Therefore, we believe that there is still a significant room for performance improvement for our scheme, especially in the verification algorithm, where the dominative cost is the GCK function. On the other hand, since we implemented the PRF functions of our scheme using Intel intrinsics, TACHYON might face a performance penalty on other platforms. Therefore, light-weight symmetric ciphers or hash functions should be preferred to implement the PRF calls in TACHYON on other platforms.

5.6 Conclusion

In this chapter, we proposed a new digital signature scheme with a post-quantum promise, which we refer to as TACHYON. Our unique algorithmic design leverages the well-known HORS construction and additively homomorphic GCK functions to extend one-time signatures to (polynomially bounded) many-time signatures. TACHYON offers several desirable properties: (i) It achieves the lowest end-to-end delay with the fastest signature generation and verification among its counterparts in every security level. (ii) TACHYON has the smallest private key size (i.e., $\kappa$-bit) among its counterparts. (iii) TACHYON has highly tunable parameters, which offer various speed and storage trade-offs. (iv)
TACHYON does not require any Gaussian sampling, and therefore it is immune to the side-channel attacks targeting this function. All these desirable properties of TACHYON come with a larger public key than most of its counterparts.
Chapter 6: Lattice-Based Proof-of-Work for Post-Quantum Blockchains

6.1 Introduction

Consensus mechanisms are at the heart of the decentralized nature of blockchains. Prevalent cryptocurrencies like Bitcoin [158] use a blockchain, as a distributed ledger, to record transactions in a consecutive manner. This distributed ledger is maintained by a peer-to-peer network of miners, who are incentivized to find a solution to a cryptographic puzzle called a Proof of Work (PoW). The first miner to find a solution can add a block of transactions to the blockchain. The more computational power a miner invests, the higher their chances of solving a puzzle first.

In open systems, such as Bitcoin, any user who is willing to dedicate a certain amount of computational resources is allowed to join the network and contribute to maintaining the blockchain. This openness, along with the incentivization strategy, has attracted large quantities of computational resources in the form of individual miners and mining pools. Currently, there are a small number of mining pools that add the majority of blocks of transactions in the Bitcoin network.\(^\text{10}\)

There has been a large array of works focusing on the security of Bitcoin’s consensus protocol. Some recent works (e.g. [102] and [167]) prove that if a majority of the computational resources are controlled by honest miners, these protocols achieve a number of useful security properties. On the other hand, these works show that if malicious entities control a majority of the computational resources in the network, then these security properties are no longer guaranteed. Recently, it has been shown that this scenario, where a mining pool exceeds 50% of the computational resources in

\(^{10}\)https://www.blockchain.com/charts/pools
Bitcoin’s peer-to-peer network, has occurred. For instance, the mining pool GHash.io had a majority of the computational resources of the Bitcoin network in 2014 [110].

6.1.1 Motivation

Proofs of Work are the most common consensus algorithms and have been adopted in over 90% of blockchains [106] (including Bitcoin [158]). In these blockchains miners must prove that they have carried out some computational work, mostly in the form of evaluating cryptographic hash functions. For example, each miner searches for a nonce $n$ such that $\text{SHA256}(c, n) \leq 2^{256-d}$, where $c$ is the hash of the previous block and $d$ is the difficulty set by the network. The hash function is repeatedly called on different nonces to find one that satisfies the above condition. The miner who satisfies this condition then broadcasts the block (including the nonce) and other nodes in the peer-to-peer network verify whether this block indeed satisfies the condition, and if so add the block to their blockchains. Thus, the consensus over the distributed ledger and decentralized trust are ensured, at the cost of terawatts of energy.\(^{11}\)

The PoW protocols in the majority of blockchain applications (e.g. Bitcoin [158]) rely on cryptographic hash functions such as SHA256. The hash functions used in such systems achieve desirable security properties against quantum adversaries when modeled as a random oracle [201]. Despite this, Grover’s search algorithm [113] gives an asymptotic quadratic advantage to quantum computers when solving hash based PoW. Given this quadratic advantage, the debut of quantum computers will increase the risk of the 51% attack [106] against hash based PoW. Therefore, while some advantage over classical computers may agree with the nature of PoW protocols (more expensive or powerful machines should perform better), we consider it a valuable research topic to reduce this

\(^{11}\text{https://digiconomist.net/bitcoin-energy-consumption/}$.}
advantage, e.g. because quantum computers may exist for some time before being available to the public.

In this work we look to the NIST standardization process,\textsuperscript{12} which aims to ensure a smooth transition to post quantum secure asymmetric cryptosystems, and apply ideas from lattice based candidates to PoW based blockchains.

6.1.2 Our Contributions.

The main goal of this paper is to address the research gap in the state-of-the-art by creating a novel consensus protocol (specifically, a PoW algorithm) that reduces the advantage of quantum computers over classical ones, has fast verification, and adjustable difficulty. To achieve this goal, we propose a new PoW protocol called LPoW. The new PoW protocol is based on the Hermite-SVP problem. Given the current understanding of SVP type problems, LPoW satisfies the following properties [3, Section IV]:

- LPoW provides little quantum advantage; the asymptotic quantum advantage against SVP is less than the quadratic speed up of Grover’s algorithm.

- LPoW is hard to solve but easy to verify. Solving is equivalent to solving Hermite-SVP to a small approximation factor. Verifying is equivalent to calculating a norm, an $n^{th}$ root, and some multiplications.

- The parameters of LPoW are easy to fine tune to adjust its difficulty. In particular increasing the dimension of the lattice has a well studied effect on the computational resources required to solve the PoW.

\textsuperscript{12}https://csrc.nist.gov/projects/post-quantum-cryptography
A secondary goal of this paper is to create a PoW protocol that encourages further experimentation with, and understanding of, practical algorithmic improvements for solving SVP type problems.

In [116], the authors suggest harnessing both the energy spent on hash puzzles, and the demand to mine cryptocurrencies, to improve the state-of-the-art in discrete log cryptanalysis. Following [116], and given that the difficulty of SVP is fundamental to the security of lattice based submissions to NIST’s post quantum standardization process, an SVP based PoW can similarly leverage this energy and demand to aid in the cryptanalysis of the SVP problem.

In Section 6.3 we discuss several areas of SVP solving strategies which could benefit from increased attention.

6.1.3 Limitations

If we assume a given hash function is a random oracle, then Grover’s algorithm gives the optimal speedup against PoW based on this hash function, and cannot be parallelized except in the trivial manner [213].

Effectively this means that the PoW parameters, e.g. the number of leading zeros required in the hash output, will only have to increase to account for increased computational strength, and not fundamentally new algorithmic techniques. This is not necessarily the case for the specific lattice problem we consider; we do not have any proofs of optimality for the algorithms currently used to solve it. In effect, this means that the PoW parameters, i.e. the lattice rank, may need to be increased to account for algorithmic improvements, as well as for increased computational strength.

We note that the best known time complexity for solving the SVP puzzles we consider is $2^{\Theta(n)}$, for lattices of rank $n$, and that any change to even slightly subexponential in $n$ would represent
a huge moment in the theory of lattices. Therefore, we do not expect to have to increase the rank too much, even to account for any algorithmic improvements.

6.2 Proposed PoW Protocol, LPoW

Proof of work protocols enable a prover to prove to a verifier that it has executed a certain amount of work. Before we present our PoW protocol, we present formally define these protocols as follows. We adopt the definition of such protocols from [24], which consists of algorithms that generate a challenge, solve such a challenge, thereby producing a proof of solution, and finally verify that this proof is correct. This triple of algorithms must satisfy the following.

Definition 21. A \((t(n), \delta(n))\)-Proof of Work (PoW) consists of three algorithms \((\text{Gen}, \text{Solve}, \text{Verify})\) that satisfy the following.

- **Efficiency:**
  - \(\text{Gen}(1^n)\) runs in time \(\tilde{O}(n)\).
  - For any \(c \leftarrow \text{Gen}(1^n)\), \(\text{Solve}(c)\) runs in time \(\tilde{O}(t(n))\).
  - For any \(c \leftarrow \text{Gen}(1^n)\), \(\Pi \leftarrow \text{Solve}(c)\), \(\text{Verify}(c, \Pi)\) runs in time \(\tilde{O}(n)\).

- **Completeness:** For any \(c \leftarrow \text{Gen}(1^n)\) and any \(\Pi \leftarrow \text{Solve}(c)\),

\[
\Pr[\text{Verify}(c, \Pi) = \text{acc}] = 1,
\]

with the probability taken over the randomness of \(\text{Verify}\).\(^\text{13}\)

\(^\text{13}\)We note that our \(\text{Verify}\) is deterministic.
- **Hardness**: For any polynomial \( l \), any constant \( \epsilon > 0 \), and any algorithm \( \text{Solve}_i^* \) that runs in time \( l(n)t(n)^{1-\epsilon} \) when given as input \( l(n) \) challenges \( \{c_i \leftarrow \text{Gen}(1^n)\}_{i \in [l(n)]} \),

\[
\Pr[\text{Verify}(c_i, \Pi_i) = \text{acc}, \forall i \mid (\Pi_1, \ldots, \Pi_{l(n)}) \leftarrow \text{Solve}_i^*(c_1, \ldots, c_{l(n)})] < \delta(n)
\]

with the probability taken over the randomness of \( \text{Gen} \) and \( \text{Verify} \).

Efficiency ensures that verification runs in (near) linear time. Efficiency and completeness together ensure that a prover that performs roughly \( t(n) \) operations can prove to the verifier that it has done so. Hardness requires that the prover has, e.g. a negligible chance, for \( \delta \) some negligible function of \( n \), to convince the verifier without performing \( l(n)t(n) \) operations. This remains true, even if the prover may compute on the \( l(n) \) challenges together.

Before we propose our new PoW protocol, we give a brief précis of how instances of Hermite-SVP problems are solved. One can solve SVP on \( \Lambda \) using a variety of families of algorithms. The family we consider is heuristic lattice sieves, which have the best known classical and quantum time complexity, standing at \( 2^{0.292n+\omega(n)} \) [26] and \( 2^{0.265n+\omega(n)} \) [134] respectively.

However, it is not necessary to call lattice sieves in the full dimension of the lattice to solve SVP type problems [83]. Instead sieving in dimension \( n - \Theta(n/\log n) \) suffices under certain heuristic assumptions.

There also exist many further heuristic techniques that provide significant practical speedups [136, 196]. Finally, a framework that collates, extends, and implements these techniques holds the record for the highest dimension SVP challenge solved [10, 91].

The techniques mentioned above depend non trivially on the “quality” of the lattice basis being used, informally; how short and close to orthogonal its basis vectors are. Therefore lattice
reduction algorithms such as BKZ are employed [74, 184], which themselves require SVP oracles for lower dimensional projected sublattices. The constant suppressed in the $\Theta(n/\log n)$ above will depend on the methods used to improve the quality of the basis. Some experimental values may be found in [10, Fig. 3b].

The high level design of our PoW follows Definition 21. We set $n$ as the dimension of the lattice and let $\alpha, p$ follow the Darmstadt SVP Challenges.

**Definition 22.** Let LPoW be defined by the following triple $(\text{Gen}, \text{Solve}, \text{Verify})$.

- **Gen**$(1^n; r)$, let the randomness $r$ be explicit and derived from the previous block. First, sample a prime $p$ of bitsize $10n$, then sample sample i.i.d. uniform $x_2, \ldots, x_n \leftarrow U(\{0\} \cup [p - 1])$, to form a basis $B$ as in Equation 2.1. Let $\alpha = 1.05 \cdot \Gamma(n/2 + 1)^{1/n}/\sqrt{\pi}$. Return $c = (\alpha, n, B, p)$.

- **Solve**$(c)$, the miner parses $c$ as $(\alpha, n, B, p)$ and attempts to find a vector $v \in \Lambda(B) \setminus \{0\}$ such that $\|v\| \leq \alpha \cdot p^{1/n}$. It outputs $\Pi = (v, \ast)$, where $v = B \cdot \ast$.

- **Verify**$(c, \Pi)$, parses $c$ as $(\alpha, n, B, p)$ and $\Pi$ as $(v, \ast)$, and outputs

$$\text{acc} = \|v\| \leq \alpha \cdot p^{1/n} \land v = B \cdot \ast \land \ast \in \mathbb{Z}^n.$$  

We use an extendable output function, e.g. [92], to extract sufficient randomness from the previous block to sample the required quantities in Gen. In the following, we weaken ever so slightly the efficiency requirements of Gen and Verify.

For Gen it is not known how to generate an $n$ bit prime, either probably or provably, in $\tilde{O}(n)$. Indeed, the prime number theorem tells us that an $n$ bit odd number is prime with probability
approximately $1/n$ and no known primality test runs in polylog($n$). Instead, by using the Miller–Rabin test [176] with $O(n)$ random bases on uniform odd $n$ bit integers, we may generate a probable $n$ bit prime in expected time $O(n^3 \cdot m(n))$ [98, Thm 12.2.2]. As verify requires the multiplication of a matrix and a vector, it costs $O(n^2 \cdot m(n))$.

Theorem 5. Let $t_c(x) = 2^{0.292x + o(x)}$ and $t_q(x) = 2^{0.265x + o(x)}$, and $\delta(n)$ be a negligible function of $n$, then, under current SVP solving techniques, there exists an $x(n) \in n - \Theta(n/\log n)$ such that LPoW is a $(t_c(x(n)), \delta(n))$ PoW against classical computers, and a $(t_q(x(n)), \delta(n))$ PoW against quantum computers.

Proof. We may generate a probable $10n$ bit prime in expected time $O(n^3 \cdot m(n))$, and $n - 1$ samples from $U(\{0\} \cup [p-1])$ in time $O(n \log n)$, and hence a challenge $c$.

The most efficient known algorithms Solve on input a challenge $c$ call at least one, and at most poly($n$), SVP oracles in dimension in $x(n) \in n - \Theta(n/\log n)$ [10, 83]. Therefore in the classical case $t(n) = t_c(x(n))$, and in the quantum case $t(n) = t_q(x(n))$, using the most efficient known classical and quantum SVP oracles.

Note that $n - cn/\log n \in \Theta(n)$ for any constant $c$, and while we do not prove that the SVP oracle must be called in dimension $x(n) \in \Theta(n)$, any $x(n) \in o(n)$ would imply a subexponential time algorithm for our $\alpha$-HSVP problem, and therefore for $\alpha^2$-approx-SVP [144]. As $\alpha^2 \in O(n)$, this would be a major breakthrough. Verifying a solution to a challenge can be performed in time $O(n^2 \cdot m(n))$. This concludes the discussion on efficiency.

We expect $1.05^n$ solutions for a challenge, for large $n$, and therefore the PoW is complete with all but negligible probability. To make it perfectly complete one may take instead $\alpha' = \sqrt{2} \cdot \Gamma(n/2 + 1)^{1/n}/\sqrt{\pi}$ and set $n$ larger as appropriate to maintain a desired cost for Solve.
Finally, it is not known how to use information from independent random lattices as advice for Hermite-SVP problems in other random lattices. Given \( l(n) \in \text{poly}(n) \) lattices generated by \( \text{Gen}(1^n) \) the probability, under the Gaussian heuristic, that any of the them share a sufficiently short vector is in \( \text{poly}(n) \cdot 1.05^n/p \in \text{negl}(n) \). Without knowing how to otherwise use advice from other lattices, we therefore have \( \delta(n) \in \text{negl}(n) \).

6.3 Discussion

We calculate a value of \( n \) that we expect to very roughly match the current cost of mining a bitcoin, 21.45 terahashes.\(^\text{14}\)

Assuming SHA-256, on input 64 bytes, takes approximately 1500 cycles, this gives approximately \( 2^{55} \) cycles. The top few data points of [10, Table 2], which uses identically generated random lattices, have dimensions 151, 153, 155 and approximate cycle counts \( 2^{56}, 2^{57}, 2^{57} \) respectively. Therefore we suggest \( n \geq 150 \), at least given current methods. The recent work of [91] uses GPU cores to the same challenges up to \( n = 180 \), and [91, Table 1] gives another set of experimental values against which to parameterize the necessary difficulty.

We list here topics that could benefit from the attention \( \text{LPoW} \) may bring to Hermite-SVP. As mentioned in Section 3.3, heuristic techniques for solving SVP, e.g. the amount of attainable “dimensions for free” \( \Theta(n/\log n) \), depend on the quality of the lattice basis. Clearly, the hidden constant is important. In [10, 83] some analyses of attainable dimensions for free are given. However, given the public availability of G6K,\(^\text{15}\) a more thorough survey of how the variants of BKZ, the insertion scoring functions, and the sequences of instructions e.g. Pump and WorkOut, described therein, affect these dimension saving techniques is possible.

\(^{14}\)https://btc.com/stats/diff, retrieved 2021/03/06.
\(^{15}\)https://github.com/fplll/g6k.
A downside of sieving is the exponential memory cost, which may lead to memory access delays that become a bottleneck. It has been suggested that this could be partially mitigated by hardware implementations of sieves [84, 131]. Given the enormous resources put into developing ASICs for hash based PoW, one may expect similar advances to be feasible in the case of LPoW, as well as advances beyond the parallelism offered by G6K [10, App B]. In particular, one may hope for advances upon previous work on distributed sieving [56] to larger or more general contexts.

Finally, recent works on concrete quantum circuits and the application of error correction estimate the speedups attainable in practice from quantum search when used in the context of hash functions [16] and lattice sieves [11]. While the cited works suggest that, under our current understanding of quantum computers, little to no advantage would be gained from the use of a quantum computer when solving PoW today, we are considering the case where e.g. improvements in classical computational power push the required hardness of PoW into ranges where a quantum computer would provide a meaningful advantage, or where more efficient error correction of quantum circuits is available. At worst, we have specified a new PoW based on well studied hard problems. This work ultimately derives from our desire to create a PoW that future proofs blockchains against giving a large advantage to quantum computers.

6.4 Related Work

Blockchain technology is achieved by leveraging a set of symmetric and asymmetric cryptographic primitives, which are vital for providing distributed trust, transaction authentication, privacy, and so forth. Hash functions in particular are essential for establishing the continuity between blocks in the blockchain. However, all of these primitives, to varying degrees, are threatened by the eventual emergence of quantum computers.
There have been numerous efforts to devise and standardize post-quantum secure cryptographic primitives. For example, NIST are expected to produce the first draft of their post-quantum standard before 2024. Similarly, some research has focused on post-quantum blockchain systems. Bitcoin Post-Quantum was proposed as an experimental branch of Bitcoin. Bitcoin Post-Quantum employs XMSS [66] signatures to replace the ECDSA [15] signatures that are currently deployed on Bitcoin’s blockchain. XMSS is a hash-based signature scheme that, while it provides a solid post-quantum security promise, suffers from large parameters. Abelian\textsuperscript{16} is investigating the use of more efficient post-quantum signature schemes such as lattice-based schemes to replace the conventional signature schemes currently used in blockchains. Ethereum\textsuperscript{17} 3.0 has future plans to deploy post-quantum secure protocols such as Zero-Knowledge Scalable Transparent Arguments of Knowledge (zk-STARKs) which enjoy post-quantum security.

There have been also some PoW protocols that strive to attain a better post-quantum security promise when compared to conventional PoW protocols. These approaches mainly look into solving memory-intensive problems. The first instance of these protocols is called Proof-of-Space [93] in which the prover generates a memory-hard function and the verifier asks for a subset of memory locations to check if they are filled with a proper function. While the protocol enjoys an efficient verification, the memory-hard core of the scheme, based on superconcentrators, is rather slow. Based on the reports in [166], it takes around one minute to fill 1GB of memory.

The Cuckoo cycle PoW [199] requires the prover to find a cycle of a particular length in a directed bipartite graph consisting of $N$ vertices and $O(N)$ edges. The original scheme was broken in [17] by reducing the memory requirement by a factor of 50 with a $2\times$ increase in computational cost. Another scheme called Momentum was proposed in [143] which searches for collisions in 50-bit outputs of a hash function with 26-bit input. While no detailed time-space trade-off is provided\textsuperscript{16}\textsuperscript{17}.

\textsuperscript{17}https://docs.ethhub.io/ethereum-roadmap/layer-2-scaling/zk-starks/
in [143], minimizing the memory requirement by a factor of $q$ only incurs a $\sqrt{q}$ factor increase in additional running time [202]. Lastly, there is a recent work, called Equihash [49] which is based on the generalized birthday problem. To make an algorithm resistant to amortization, the authors in [49] propose a technique called algorithm binding by utilizing the properties of Wagner’s algorithm [205].

As mentioned, to our knowledge LPoW is the first computational PoW protocol which strives to minimize the gap between a classical and quantum miner.
Chapter 7: Lattice-Based Public Key Searchable Encryption

7.1 Introduction

Cloud computing has significantly impacted the computing infrastructure and enabled a large pool of applications. For example, data outsourcing [18] permits small/medium-sized businesses to increase data availability by minimizing the management and maintenance costs. Data outsourcing, despite its merits, raises significant data privacy concerns for clients. Traditional encryption techniques can be used to overcome such privacy concerns. However, standard encryption does not permit search capabilities on the encrypted data. Therefore, a significant amount of research is focused on Searchable Encryption (SE) technologies that can be used to efficiently address this problem. There are two main branches of SE where each is tailored for a distinct set of applications.

Dynamic Searchable Symmetric Encryption (DSSE) (e.g., [191, 212]) provides search capabilities on encrypted data for private data outsourcing applications (e.g., data storage on the cloud), in which the client uses her own private key to encrypt and then search on her own data over the cloud. Among other symmetric key based methods, Oblivious Random Access Machine (ORAM) [108, 117] is used to hide user’s access patterns. Public Key Encryption with Keyword Search (PEKS) schemes [52] allow any client to encrypt data with specified keywords under the public key of a designated receiver. The designated receiver, Alice, can then use her private key to generate and send trapdoors for her desired keywords, and enable the server to search on the encrypted data to retrieve the files that are associated with the keyword. PEKS is well suited for distributed applications (e.g., e-mail, audit logging for Internet of Things, etc.) where a large number

\[\text{This chapter was published in [28, 32]. Permission is included in Appendix A.}\]
The focus of this paper is on PEKS schemes.

Figure 7.1 illustrates a potential application which is considered as the main motive for the initial proposal of PEKS schemes in [52]. Alice, who is assumed to have a number of devices (e.g., cellphone, desktop, etc.), wants her e-mails to be routed to her devices based on the keywords associated with them. For instance, when the sender, Bob, sends her an e-mail with keyword “urgent”, the e-mail should be routed to her cellphone. To achieve this, after encrypting the e-mail content with a conventional public key encryption, Bob uses a PEKS scheme to encrypt the keyword “urgent” and sends it together with the encrypted e-mail to the e-mail server. Alice can then use her private key to generate the trapdoor corresponding to keyword “w = urgent” and ask the server to retrieve all the e-mails associated with w.

Another important application of PEKS schemes is in storing and searching on private log files. PEKS schemes can enable a heterogeneous set of devices to send encrypted log files concatenated with a searchable ciphertext of distinct keywords to an untrusted storage server. To analyze the log files, an auditor can use his private key to generate trapdoors and enable the server to search and return the files that are associated with the target keyword.
7.1.1 Research Gap

Since their introduction in [52], several PEKS schemes with a variety of features have been proposed (e.g., [54, 65, 165, 187]). However, the wide adoption of PEKS schemes in practice has been hindered due to a number of obstacles:

- **High End-to-End Delay:** The most computationally expensive part of the PEKS is generally the search phase, which requires the execution of a Test algorithm for each keyword-file pair in the database. The existing PEKS schemes (e.g., [52, 54]) introduce a significant end-to-end computational delay due to at least one pairing operation that is required in the Test algorithm which is to be called linear to the total number of keyword-file pairs in the database, for each search query. Therefore, providing a computational efficient Test algorithm is a critical requirement to minimize the end-to-end delay.

- **Lack of Long-term Security:** It is a highly desirable property for data storage applications to offer long-term data security measures. However, achieving a high level of security for an extended duration of time requires a continuous increment of key sizes, which results in a substantial increase in computation overhead for conventional cryptographic primitives (e.g., ECC, RSA). Furthermore, the emergence of quantum computers will render most of the conventional asymmetric cryptography primitives unsafe, and therefore, there is a great merit in devising PEKS schemes that can provide post-quantum security promises.

- **Lack of Full-Fledged Implementations:** While a number of DSSE schemes have been fully implemented on real data (and are publicly accessible, e.g., [59, 60]), to the best of our knowledge, no full-fledged implementation (with a real dataset) of PEKS schemes is publicly available to this date. Hence, there is a need for providing a full-fledged implementation of PEKS schemes and benchmarking their deployment on actual cloud platforms to measure
important performance factors (e.g., communication delay, disk access time) that cannot be precisely captured with mere "cost estimations".

7.1.2 Our Contribution

The main goal of this paper is developing a significantly more efficient PEKS scheme and proposing a PEKS scheme with worst-case reduction by harnessing the existing lattice-based IBE schemes using Abdalla et. al. generic anonymous-IBE to PEKS transformation. We also provide an extensive analysis to highlight the advantages and disadvantages of such schemes.

Towards addressing the aforementioned research gaps, we developed two lattice-based PEKS schemes with a post-quantum security promise and presented a full-fledged implementation of our efficient lattice-based PEKS scheme and its pairing-based counterpart. We outline our contributions as follows:

- **First Lattice-Based PEKS Schemes**: In the initial proposal of PEKS in [52], Boneh et al. showed how to derive a PEKS scheme from an Identity-Based Encryption (IBE). Abdalla et al. [2] specified the requirements for the underlying IBE scheme to ensure the security and consistency of the derived PEKS. In this paper, we propose two PEKS schemes based on lattices-based tools. (i) Our first scheme is referred to as NTRU-PEKS, which harnesses Ducas et al.’s IBE scheme [88], and it meets the all requirements provided by [2] to ensure provable security in Random Oracle Model (ROM). (ii) Our second scheme is referred to as LWE-PEKS, which leverages IBE constructions in [4, 62] to offer the first provable secure lattice-based PEKS in the standard model (to the best of our knowledge). We prove the security and consistency of our PEKS schemes and suggest parameter sizes to avoid potential consistency errors (with an overwhelming probability).
• **High Computational Efficiency**: Our NTRU-PEKS scheme offers significant computational efficiency advantages over the existing PEKS schemes. This is achieved by harnessing the latest efforts in improving the efficiency of the lattice-based schemes, ring-LWE [177] and fast arithmetic operations (e.g., Fast Fourier Transform) over polynomial rings $\mathbb{Z}[x]/(x^N + 1)$. As it is shown in Table 7.2, the NTRU-PEKS scheme has significantly more efficient Test and PEKS algorithms than those in [52, 214], which are currently considered as the most efficient PEKS alternatives [58]. The efficiency of the Test algorithm is of vital importance, since it is executed by the server once per every keyword-file pair in the database which results in $O(L)$ computation overhead, where $L$ is the total number of keyword-file pairs. The efficiency of the PEKS algorithm facilitates the implementation of PEKS schemes on battery-limited devices. Due to its computational efficiency, despite having larger parameters, we showed the deployment of NTRU-PEKS on actual cloud achieves a superior end-to-end response time as compared to its counterparts (see §7.4).

• **Long-term Security**: Both of our constructions are based on lattice-based tools that provide long-term security and are currently considered secure against quantum computers. It is worth noting that lattice-based schemes also have a substantially smoother performance response to increased key sizes compared to conventional cryptographic primitives (e.g., ECC, RSA).

• **Full-Fledged Implementation**: We provide a full-fledged implementation of our NTRU-PEKS scheme and its most efficient pairing-based counterpart with deployment in actual cloud setting with Enron e-mail dataset. We chose Amazon Web Server (AWS) as the server in our system, a commodity hardware and an ARM Cortex-A53 as the client machines. One can easily see the importance of a full-fledged implementation when comparing the benchmark provided from the simulation results in [32] and the benchmark results of the implementation in §7.4. Detailed
experimental results are further explained in §7.4. We also open-sourced our implementations for public testing and wide adoption (see §7.4).

7.1.3 Limitations

We present two lattice-based schemes, one with security in the random oracle model and one in the standard model. In this section, we point out their limitations as compared to their conventional pairing-based counterparts.

The Trapdoor algorithm of NTRU-PEKS (secure in the random oracle model) is slower than that of BCOP. One should note that the Trapdoor algorithm is to be executed once per each query on the receiver’s side and has negligible effect on the end-to-end delay (e.g., see Figure 7.2). Another downside of NTRU-PEKS is the parameter sizes, for instance, the searchable ciphertext size in NTRU-PEKS is significantly larger than that of BCOP [52]. We should note that while the storage blow-up remains a concern, the increased communication overhead and disk access time have negligible effects on the end-to-end delay (e.g., see Figure 7.2). NTRU-PEKS provides very efficient PEKS and Test algorithms. The efficient PEKS algorithm provides significant efficiency for the sender as it is called to generate searchable ciphertext for each keyword to be attached to the file. The Test algorithm is the main factor in reducing end-to-end delay in PEKS schemes since it is executed once for every keyword-file pairs in the database (e.g., see Figure 7.2). Given all the computation efficiency gains, we believe the storage blow-up might be a favorable trade-off for certain applications where end-to-end delay and long-term security are more critical than the storage.

In the line of proposing PEKS schemes in the standard model, to the best of our knowledge, the proposed LWE-PEKS scheme provides the highest security promise with reduction to worst-case problems. However, as shown in Section §7.4, it is significantly less efficient (in both computation
and storage) than Zhang and Imai’s pairing-based PEKS scheme which is also secure in the standard model [214].

7.2 Background

In this section, we first define some cryptographic tools that are utilized to devise the new schemes, and then define the notion of PEKS schemes and their corresponding security notions. For the sake of compliance, we try to use similar notation as in [88] and [4].

7.2.1 Identity-Based Encryption

We refer to the definition of IBE encryption scheme in Definition 13 in §4.2.

Halevi in [114] introduced a condition for an IND-CPA encryption scheme to offer the notion of anonymity (ANO-CPA). This condition requires that given two public keys $pk_0$ and $pk_1$ and a ciphertext $c$, encrypted under $pk_b$, for $b \in \{0, 1\}$, a computationally unbounded adversary should have a negligible advantage in determining $b$. Later, Abdalla et. al. [2] extended this condition to identity-based encryption schemes by including the handling of random oracles and weakening the statistical requirement to a computational one. The following definition defines the new anonymity under chosen plaintext attack for an IBE scheme (IBE-ANO-RE-CPA).

Definition 23. For an IBE scheme, given $H$ as a random oracle, and the $\text{KeyQuery}(\cdot)$ oracle (defined in Algorithm 8) we associate a bit $b \in \{0, 1\}$ to the adversary $A$ in the experiment $Exp_{\text{IBE-ANO-RE-CPA}-b}(1^k)$ given in Algorithm 8.

$A$’s advantage in this experiment is defined as:

$$\text{Adv}_{\text{IBE-ANO-RE-CPA}}(1^k) = \Pr[Exp_{\text{IBE-ANO-RE-CPA}-1}(1^k) = 1] - \Pr[Exp_{\text{IBE-ANO-RE-CPA}-0}(1^k) = 0]$$
Algorithm 8 Anonymity under chosen plaintext attack (IBE-ANO-RE-CPA)

KeyQuery(id)
1: idSet ← idSet ∪ id
2: return sk

Exp_{IBE,A}^{IBE-ANO-RE-CPA-b}(1^k)
1: idSet ← ∅, (mpk,msk) ← Setup(1^k)
2: (id_0, id_1, m) ← A^{KeyQuery(·),H}(mpk) : find stage
3: c ← Enc^H(m, id_b, mpk)
4: b' ← A^{KeyQuery(·),H(c)} : guess stage
5: if \{id_0, id_1\} ∩ idSet = ∅ return b' else, return 0

After queries to KeyQuery(·) and random oracle H in the find stage, A outputs a challenge (id_0, id_1, m). A will then receive c, which is an encryption of the message m under id_b where b ∈ \{0, 1\} is the output of a fair coin flip. In the guess stage, given c, A will output its decision bit b' and wins if b' = b where id_0 and id_1 were never queried to the KeyQuery(·) oracle.

Lemma 4. If an IBE scheme is IBE-IND-CPA and IBE-ANO-RE-CPA-secure, then it is also IBE-ANO-CPA-secure.

Proof. Please refer to [2].

7.2.1.1 Abdalla et. al. IBE to PEKS transformation [2]

The authors prove that if an IBE scheme is IBE-ANO-CPA (in the sense of [114]) and IBE-ANO-CPA, then one can obtain a secure PEKS scheme. We use this transformation to obtain NTRU-PEKS and LWE-PEKS.

7.2.2 Public Key Encryption with Keyword Search

A PEKS scheme consists of the following algorithms.

Definition 24. A PEKS scheme is a tuple of four algorithms PEKS = (KeyGen, PEKS, Trapdoor, Test) defined as follows.
\((pk, sk) \leftarrow \text{PEKS.KeyGen}(1^k)\): On the input of the security parameter(s), this algorithm outputs the public and private key pair \((pk, sk)\).

\(s_w \leftarrow \text{PEKS.SEnc}(pk, w)\): On the input of user’s public key \(pk\) and a keyword \(w \in \{0, 1\}^*\), this algorithm outputs a searchable ciphertext \(s_w\).

\(t_w \leftarrow \text{PEKS.Trapdoor}(sk, w)\): On the input of a user’s private key \(sk\) and a keyword \(w \in \{0, 1\}^*\), this algorithm outputs a trapdoor \(t_w\).

\(d \leftarrow \text{PEKS.Test}(t_w, s_w)\): On the input of a trapdoor \(t_w = \text{Trapdoor}(sk, w')\) and a searchable ciphertext \(s_w = \text{SEnc}(pk, w)\), this algorithm outputs a decision bit \(d = 1\) if \(w = w'\), and \(d = 0\) otherwise.

Keyword indistinguishability against an adaptive chosen-keyword attack for a PEKS scheme (PEKS-IND-CKA) is defined in the following experiment.

**Definition 25.** Given a PEKS scheme, a random oracle \(H\), and the \(\text{TdQuery}(\cdot)\) oracle as defined in Algorithm 9, we associate a bit \(b \in \{0, 1\}\) to the adversary \(A\) in the experiment \(\text{Exp}_{\text{PEKS-IND-CKA}}^{b}(1^k)\) given in Algorithm 9.

**Algorithm 9** Keyword indistinguishability against an adaptive chosen-keyword attack (PEKS-IND-CKA)

\[
\begin{align*}
\text{TdQuery}(w) \quad & 1: wSet \leftarrow wSet \cup w \\
2: t_w \leftarrow \text{Trapdoor}(w, sk, pk) \\
3: \text{return } t_w \\
\text{Exp}_{\text{PEKS, } A}^{\text{PEKS-IND-CKA}}(1^k) \quad & 1: wSet \leftarrow \emptyset, (pk, sk) \leftarrow \text{KeyGen}(1^k) \\
2: (w_0, w_1) \leftarrow A^{\text{TdQuery}(\cdot), H}(pk) \quad & \text{: find stage} \\
3: s_w \leftarrow \text{SEnc}_H(pk, w_b) \\
4: b' \leftarrow A^{\text{TdQuery}(\cdot), H}(s_w) \quad & \text{: guess stage} \\
5: \text{if } \{w_0, w_1\} \cap wSet = \emptyset, \text{return } b' \text{ else, return } 0
\end{align*}
\]
A’s advantage in the above experiment (Algorithm 9) is defined as:

\[
Adv_{PEKS,A}^{PEKS-IND-CKA}(1^k) = \Pr[Exp_{PEKS,PEKS-IND-CKA-1}^{PEKS,PEKS,1}(1^k) = 1] - \Pr[Exp_{PEKS,PEKS-IND-CKA-0}^{PEKS,PEKS,0}(1^k) = 0]
\]

After queries to \(T_{dQuery}(\cdot)\) and hash oracle \(H\), in the \textit{find} stage, \(A\) outputs a challenge \((w_0, w_1)\). \(A\) will then receive \(s_w\), which is a searchable encryption of \(w_b\) where \(b \in \{0, 1\}\) is the output of a fair coin flip, under \(pk\). In the \textit{guess} stage, given \(s_w\), \(A\) will output its decision bit \(b'\) and wins if \(b' = b\) where \(w_0\) and \(w_1\) were never queried to the \(T_{dQuery}(\cdot)\) oracle.

Due to the properties of lattice-based encryption schemes, and following the work of [81], we investigate the consistency of lattice-based PEKS schemes \(s\) from two aspects, namely, right-keyword consistency and adversary-based consistency [2].

- **Right-Keyword Consistency**: Right-keyword consistency implies the success of a search query to retrieve records associated with keyword \(w\) for which the PEKS algorithm had computed a searchable ciphertext. More specifically, right-keyword consistency refers to the decryption error in the underlying IBE scheme which leads to the inconsistency of the \texttt{Test} algorithm in the resulting PEKS scheme.

- **Adversary-Based Consistency**: We define the adversary-based consistency [2] in the following definition.

\textbf{Definition 26.} For a random oracle \(H\), the adversary-based consistency of a PEKS scheme is defined in Algorithm 10 via experiment \(Exp_{PEKS,PEKS-Consist}^{PEKS,PEKS}(1^k)\).

\(A\)’s advantage in the above experiment is \(Adv_{PEKS,A}^{PEKS-Consist}(1^k) = \Pr[Exp_{PEKS,PEKS-Consist}^{PEKS,PEKS}(1^k) = 1]\).

We note that for schemes with security in the standard model, the random oracle \(H\) is eliminated in Definition 23, Definition 25 and Definition 26.
Algorithm 10 Adversary-based consistency of a PEKS scheme (PEKS-Consist)

\[ \text{Exp}^{\text{PEKS-Consist}}_A(1^k) \]
1: \((pk, sk) \leftarrow \text{KeyGen}(1^k)\)
2: \((w_0, w_1) \leftarrow A^H(pk)\)
3: \(s_{w_0} \leftarrow \text{PEKS}^H(pk, w_0)\)
4: \(t_{w_1} \leftarrow \text{Trapdoor}^H(pk, w_1)\)
5: if \(w_0 \neq w_1\) and \([\text{Test}^H(pk, t_{w_1}, s_{w_0}) = 1]\), return 1 else, return 0

7.3 Proposed Schemes

In this section, we first propose our scheme based on NTRU lattices (i.e., NTRU-PEKS) which enjoys from highly efficient Test and PEKS algorithms and then put forth our scheme in the standard model (i.e., LWE-PEKS) which provides a high level of security.

7.3.1 PEKS Scheme from NTRU Lattices

In this section, we present our highly efficient NTRU-PEKS scheme using Abdalla et. al. transformation [2] to transform Ducas et. al. IBE scheme [88]. For this transformation to work, aside being IND-CPA, the underlying IBE scheme should be anonymous in the sense of Definition 23. We show that Ducas et. al.’s IBE scheme is indeed anonymous via Theorem 7. Our NTRU-PEKS scheme consists of the following algorithms.

\((pk, sk) \leftarrow \text{KeyGen}(q,N)\): Given a power-of-two integer \(N\) and a prime \(q\), this algorithm works as follows.

1. Compute \(\sigma_f \leftarrow 1.17\sqrt{\frac{q}{2N}}\) and select \(f,g \leftarrow \mathcal{D}_{N,\sigma_f}\) to compute \(\left\| \tilde{B}_{f,g} \right\|\) and \(\text{Norm} \leftarrow \max(\left\| (g, -f) \right\|, \left\| (\frac{qf^2}{f^2 + g^2}, \frac{qg^2}{f^2 + g^2}) \right\|)\). If \(\text{Norm} < 1.17\sqrt{q}\), proceed to the next step. Otherwise, if \(\text{Norm} \geq 1.17\sqrt{q}\), this process is repeated by sampling new \(f\) and \(g\).
2. Using extended euclidean algorithm, compute $\rho_f, \rho_g \in \mathbb{R}$ and $R_f, R_g \in \mathbb{Z}$ such that $\rho_f \cdot f = R_f \mod (x^N + 1)$ and $\rho_g \cdot g = R_g \mod (x^N + 1)$. Note that if $\gcd(R_f, R_g) \neq 1$ or $\gcd(R_f, q) \neq 1$, start from the previous step by sampling new $f$ and $g$.

3. Using extended euclidean algorithm, compute $u, v \in \mathbb{Z}$ such that $u \cdot R_f + v \cdot R_g = 1$. Compute $F \leftarrow q \cdot v \cdot \rho_g, G \leftarrow q \cdot u \cdot \rho_f$ and $k \leftarrow \left\lfloor \frac{F \cdot \bar{f} + G \cdot \bar{g}}{f \cdot \bar{f} + g \cdot \bar{g}} \right\rfloor \in \mathbb{R}$ and reduce $F$ and $G$ by computing $F \leftarrow F - k \cdot f$ and $G \leftarrow G - k \cdot g$.

4. Finally, compute $h = g \cdot f^{-1} \mod q$ and $B = \begin{pmatrix} \mathcal{A}(g) & -\mathcal{A}(f) \\ \mathcal{A}(G) & -\mathcal{A}(F) \end{pmatrix}$ and output $(pk \leftarrow h, sk \leftarrow B)$.

$s_w \leftarrow \text{PEKS}(pk, w)$: Given cryptographic hash functions $H_1 : \{0, 1\}^* \rightarrow \mathbb{Z}_q^N$ and $H_2 : \{0, 1\}^N \times \{0, 1\}^N \rightarrow \mathbb{Z}_q^N$, the receiver’s public key $pk$ and a keyword $w \in \{0, 1\}^*$ to be encrypted, the sender performs as follows.

1. Compute $t \leftarrow H_1(w)$ and pick $r, e_1, e_2 \leftarrow \{-1, 0, 1\}^N$, $k \leftarrow \{-1, 0, 1\}^N$.

2. Compute $c_0 \leftarrow r \cdot h + e_1 \in \mathbb{R}_q$ and $c_1 \leftarrow r \cdot t + e_2 + \left\lfloor \frac{q}{2} \right\rfloor \cdot k \in \mathbb{R}_q$.

3. Finally, the algorithm outputs $s_w = \langle c_0, c_1, H_2(k, c_1) \rangle$.

t_w \leftarrow \text{Trapdoor}(sk, w)$: Given the receiver’s private key $sk$, and a keyword $w \in \{0, 1\}^*$, the receiver computes $t \leftarrow H_1(w)$ and using the sampling algorithm $\text{Gaussian-Sampler}(B, \sigma, (t, 0))$, samples $s$ and $t_w$ such that $s + t_w \cdot h = t$.

d \leftarrow \text{Test}(pk, t_w, s_w)$: On the input of a receiver’s public key $pk$, a trapdoor $t_w$ and a searchable ciphertext $s_w = \langle c_0, c_1, H_2(k, c_1) \rangle$, this algorithm computes $y \leftarrow \left\lceil \frac{c_1 - c_0 + tw}{q/2} \right\rceil$ and outputs $d = 1$ if $H_2(y, c_1) = H_2(k, c_1)$ and $d = 0$, otherwise.
7.3.1.1 Completeness and Consistency

In this section, we show the completeness and consistency of NTRU-PEKS.

Lemma 5. Given a public-private key pair \((h, B) \leftarrow \text{KeyGen}(q, N)\), a searchable ciphertext \(s_w \leftarrow \text{PEKS}(pk, w)\), and a trapdoor generated by the receiver \(t_w \leftarrow \text{Trapdoor}(sk, w)\) our proposed scheme is complete.

Proof. To show the completeness of our scheme for \(s_w = \langle c_0, c_1, H_2(k, c_1) \rangle\), the Test algorithm should return 1 when \(\left\lfloor \frac{c_1 - c_0 \cdot t_w}{q/2} \right\rfloor = k\). To affirm this, we work as follows.

\[
    c_1 - c_0 \cdot t_w = (r \cdot t + e_2 + \frac{q}{2} k) - (r \cdot h + e_1) \cdot t_w \in \mathcal{R}_q
    = r \cdot s + e_2 + \frac{q}{2} k - t_w \cdot e_1
\]

Given \(r, e_1, e_2, t_w\) and \(s\) are all short vectors (due to the parameters of our sampling algorithm), all the coefficients of \(r \cdot s + e_2 - t_w \cdot e_1\) will be in \((-\frac{q}{4}, \frac{q}{4})\), and therefore, \(\left\lfloor \frac{c_1 - c_0 \cdot t_w}{q/2} \right\rfloor = k\).

To address right-keyword consistency issues related to the decryption error of encryption over NTRU lattices, we need to make sure that all the coefficients of \(z = r \cdot s + e_2 - e_1 \cdot t_w\) are in the range \((-\frac{q}{4}, \frac{q}{4})\) and \(q \approx 2^{24}\) for \(\kappa = 80\) and \(q \approx 2^{27}\) for \(\kappa = 192\). As highlighted in [175], for such large values of \(q\), the probability of decryption error is negligible with respect to the security parameter.

Theorem 6. \(A\)'s advantage in breaking the consistency of NTRU-PEKS scheme in the sense of Definition 26, after making \(q_1\) and \(q_2\) queries to \(H_1(\cdot)\) and \(H_2(\cdot)\), respectively, is:

\[
    \mathcal{A}_{\text{PEKS-Consist}}^{\text{PEKS}}(1^k) \leq \frac{(q_1 + 2)^2}{N^{2\log_2 q}} + \frac{(q_2 + 2)^2}{N^{2\log_2 q}} + \epsilon
\]
where the term $\epsilon$ is negligible in term of the security parameter $\kappa$.

Proof. Upon inputting $q$ and $N$, the challenger $C$ initiates the experiment $(h, B) \leftarrow \text{KeyGen}(q, N)$. It passes $h$ to the adversary $A$ and keeps $B$ secret.

- $(w_0, w_1) \leftarrow A^H(pk)$: $A$ sends $C$ two keywords $(w_0, w_1)$.
- $s_{w_0} \leftarrow \text{PEKS}^H(pk, w_b)$: $C$ computes $c_0 = r \cdot h + e_1$ and $c_1 = r \cdot H(w_0) + e_2 + \left\lfloor \frac{q}{2} \right\rfloor k$ for a random selection of $r, e_1, e_2 \leftarrow \{-1, 0, 1\}^N$, $k \leftarrow \{0, 1\}^N$, and sends $\langle c_0, c_1, H_2(k, c_1) \rangle$ to $A$.
- $t_{w_1} \leftarrow \text{Trapdoor}^H(pk, w_b)$: $C$ samples short vectors $s, t_w$ such that $s + t_w \cdot h = H(w_1)$ and returns $t_w$ to $A$.

$A$ wins when $w_0 \neq w_1$, and the Test algorithm outputs 1 (i.e, $H_2(k, c_1) = H_2(y, c_1)$).

Note that in the above game, $A$ wins when $w \neq w'$ and $H_2(z_1, z'_1) = H_2(z_2, z'_2)$. Let’s assume $A$ makes $q_1$ queries to $H_1$ and $q_2$ queries to $H_2$ oracles. Let $E_1$ be the event that there exists $(x_1, x_2)$ such that $H_1(x_1) = H_1(x_2)$ and $x_1 \neq x_2$ and let $E_2$ be the event that there exist two pairs $(z_1, z'_1)$ and $(z_2, z'_2)$ such that $H_2(z_1, z'_1) = H_2(z_2, z'_2)$ for $z_1 \neq z_2$ and $z'_1 \neq z'_2$. Then if $\Pr[\cdot]$ represents the probability of consistency definition, $\text{Adv}_{\text{PEKS}, A}^{\text{Consist}}(1^k) \leq \Pr[E_1] + \Pr[E_2] + \Pr[\text{Exp}_{\text{PEKS}, A}^{\text{Consist}}] = 1 \land \bar{E}_1 \land \bar{E}_2$.

Given the domain of our hash functions, the first and second terms are upper bounded by $\frac{(q_1+2)^2}{N^2 \log_2 q^2}$ and $\frac{(q_2+2)^2}{N^2 \log_2 q^2}$, respectively. For the last term, if $H_1(x_1) \neq H_1(x_2)$, then in our scheme, the probability that $B_1 = B_2$ is negligible due to the decryption error. Therefore, $H_2(y_1, B_1) \neq H_2(y_2, B_2)$, hence, the probability of the last term is also negligible.

7.3.1.2 Security Analysis

In this section, we focus on analyzing the security of the NTRU-PEKS scheme.
The security of lattice-based schemes can be determined by hardness of the underlying lattice problem (in case of NTRU-PEKS, ring-LWE). Therefore, similar to the LWE-PEKS scheme, we use the root Hermite factor to assess the security of the NTRU-PEKS scheme. According to [86], for a short planted vector \( v \) in an NTRU lattice, the associated root Hermite factor is computed as
\[
\gamma_n = \sqrt{\frac{N}{(2\pi e)^{n/2}}} \cdot \det(\Lambda)^{1/n} \cdot \|v\|
\]
Based on [75, 101], \( \gamma \approx 1.004 \) guarantees intractability and provides approximately 192-bit security.

Following Lemma 4, to establish the security of our NTRU-PEKS scheme, we need to rely on the security of the underlying IBE scheme. Ducas et al. provided the proof of IBE-IND-CPA of their scheme in [88]. Therefore, we are left to prove the anonymity of their scheme via Theorem 7.

**Theorem 7.** The IBE scheme of Ducas et al. is anonymous in the sense of Definition 23 under the decision ring-LWE problem.

**Proof.** Since the output of the PEKS algorithm of our scheme corresponds to the encryption algorithm of [149, 150], for \( A \) to determine \( s_w \) corresponds to which keyword with any probability
\[
\Pr \geq \frac{1}{2} + \epsilon,
\]
for any non-negligible \( \epsilon \), it has to solve the decision ring-LWE. Our scheme works over the polynomial ring \( \mathbb{Z}[x]/(x^N + 1) \), for a power-of-two \( N \) and a prime \( q \equiv 1 \mod 2N \). The ring-LWE based PEKS algorithm computes a pseudorandom ring-LWE vector \( c_0 = r \cdot h + e_1 \) (for a uniform \( r, e_1 \sim \{-1,0,1\}^N \)) and uses \( H(w) \) to compute \( c_1 = r \cdot H(w) + e_2 + \left\lfloor \frac{q}{2} \right\rfloor k \) that is also statistically close to uniform. Therefore, the adversary’s view of \( \langle c_0, c_1, H_2(k,c_1) \rangle \) is indistinguishable from uniform distribution under the hardness of decision ring-LWE. The pseudorandomness is preserved when \( t_w \) is chosen from the error distribution (by adopting the transformation to Hermite’s normal form) similar to the one in standard LWE [157]. □
Theorem 8. If there exists an adversary $A$ that can break PEKS-IND-CKA of NTRU-PEKS scheme as in Definition 25, one can build an adversary $F$ that uses $A$ as subroutine and breaks the security of the IBE scheme in Definition 23.

Proof. The proof works by having adversaries $F$ and $A$ initiating the find phase as in Definition 23 and Definition 25 respectively.

Algorithm $F^{\text{KeyQuery}(),H}(\text{find}, mpk)$

- $(mpk,msk) \leftarrow \text{Setup}(q,N)$: $F$ receives $mpk$ and passes it to $A$.

Algorithm $A^{T\text{dQuery}(),H}(\text{find}, pk)$

- Queries on $T\text{dQuery}()$: Upon such queries, $F$ queries $\text{KeyQuery}(.)$ which keeps a list $idSet$ maintaining all the previously requested queries and responses. If the submitted query exists, the same response is returned, otherwise, to sample short vectors $s,t_w$ the oracle uses $msk$ to run $(s,t_w) \leftarrow \text{Gaussian-Sampler}(msk,\sigma,(H(w),0))$ and passes $t_w$ to $F$. $F$ sends $t_w$ to $A$.

After the find phase, a hidden fair coin $b \in \{0,1\}$ is flipped.

Execute $(w_0,w_1) \leftarrow A^{T\text{dQuery}(),H}(\text{guess}, pk)$

- Upon receiving $(w_0,w_1)$, $F$ selects a message $m \in \{0\}^N$ and calls $\text{Enc}(m,w_0,w_1)$ that runs encryption on $(w_b,m)$ and outputs $s_w = (c_0,c_1,H_2(k,c_1))$. $F$ relays $s_w$ to $A$.

Finally, $A$ outputs its decision bit $b' \in \{0,1\}$. $F$ also outputs $b'$ as its response. Omitting the terms that are negligible in terms of $q$ and $N$, the upper bound on $\text{IND-CKA}$ of NTRU-PEKS is as follows.

$$\text{Adv}_{A}^{\text{PEKS-IND-CKA}}(q,N) \leq \text{Adv}_{F}^{\text{NTRU-IBE-ANO-CPA}}(q,N)$$
7.3.1.3 Discussion on Subfield Attacks

Subfield attacks target the presence of a subfield in ideal lattices to solve the overstretched version of the NTRU problem. Recently, there have been noticeable advancements to extend subfield attacks on ideal lattices [9, 43, 132]. However, as also shown in [9], these attacks do not affect the parameters that are used in our scheme.

7.3.2 Lattice-Based PEKS Scheme in the Standard Model

In this section, we present our highly secure LWE-PEKS scheme using Abdalla et. al. transformation [2] to transform Agrawal et. al. IBE scheme [4]. For this transformation to work, aside from being IND-CPA, the underlying IBE scheme should be anonymous in the sense of Definition 23. While the scheme proposed in [4] does provide the anonymity required by Abdalla et. al. transformation [2], in the adaptive security of the scheme, the number of adversarial queries was bounded by $q$. In other words, only the bounded form of the security of the scheme was proven in [4]. This restriction was later lifted in a more refined analysis in [62].

Similar to [4], we treat keywords as a sequence of $l$ bits $w = (b_1, \ldots, b_l) \in \{1, -1\}^l$. Before presenting the scheme in details, we review the tools that are needed for the correctness of our LWE-PEKS scheme. In [6], Ajtai illustrated how to sample a random uniform matrix (with a small Gram-Schmidt norm) $A \in \mathbb{Z}_q^{n \times m}$ with an associated basis $S_A$ of $\Lambda^\perp_q(A)$. The following theorem, defines the properties of $\text{TrapGen}$ algorithm [4, 14] which is used in the $\text{KeyGen}$ algorithm of the LWE-PEKS scheme.

$(A, S) \leftarrow \text{TrapGen}(q, n)$ : Given a prime $q$, a positive $n$ and $\delta = \frac{1}{3}$, there is a polynomial time algorithm $\text{TrapGen}(q, n)$ that outputs a pair $(A \in \mathbb{Z}_q^{n \times m}, S \in \mathbb{Z}^{m \times m})$ s.t., $A$ is statistically close to
uniform and \( \mathbf{S} \) is a basis for \( \Lambda_q^\perp(A) \), where \( m > 6n \log q \) and \( \|\hat{\mathbf{S}}\| \leq \mathcal{O}(\sqrt{n \log q}) \) and \( \|\mathbf{S}\| \leq \mathcal{O}(n \log q) \) hold with a high probability.

Following [4], we set \( \sigma_{TG} = \mathcal{O}(\sqrt{n \log q}) \) as the maximum Gram-Schmidt norm of the basis generated by \( \text{TrapGen}(q, n) \).

In the following we define the sampling algorithm which is used to generate trapdoors in our scheme (i.e., \text{SampleLeft}), the same algorithm, with identical properties has also been used in [70, 169].

\[ \mathbf{e} \leftarrow \text{SampleLeft}(A, M_1, T_A, u, \sigma) : \text{Given an } n\text{-rank matrix } A \in \mathbb{Z}_q^{n \times m}, \text{ a matrix } M_1 \in \mathbb{Z}_q^{n \times m_1}, \text{ a short basis of } \Lambda_q^\perp(A), \text{ a vector } u \in \mathbb{Z}_q^n, \text{ and a Gaussian parameter } \sigma > \|\hat{T}_A\| \cdot \omega(\sqrt{\log(m + m_1)}), \text{ this algorithm outputs a vector } \mathbf{e} \in \mathbb{Z}^{m + m_1} \text{ sampled from the distribution statistically close to } D_{\Lambda_q^\perp(F_1), \sigma} \text{ where } F_1 := (A | M_1). \]

In the following, we present the LWE-PEKS scheme in detail.

\[ (pk, sk) \leftarrow \text{KeyGen}(\lambda) : \text{On the input of the security parameter } \lambda, \text{ and the parameters } q, m, n, \sigma, \alpha \text{ (set as instructed in the following section), the receiver works as follows to generate her key pair.} \]

1. Use \( \text{TrapGen}(q, n) \) to pick a random matrix \( A_0 \in \mathbb{Z}_q^{n \times m} \) with basis \( T_{A_0} \) for \( \Lambda_q^\perp(A_0) \) s.t. \( \|T_{A_0}\| \leq \mathcal{O}(\sqrt{n \log q}) \).
2. Select \( l + 1 \) random matrices \( A_1, \ldots, A_l, B \leftarrow \mathbb{Z}_q^{n \times m} \) and a random vector \( u \leftarrow \mathbb{Z}_q^n \).
3. Output the public key \( pk \leftarrow (A_0, A_1, \ldots, A_l, B, u) \) and secret key \( sk \leftarrow T_{A_0} \).

\[ s_w \leftarrow \text{PEKS}(pk, w) : \text{On the input of the } pk \text{ and an } l\text{-bit keyword } w = (b_1, \ldots, b_l) \in \{1, -1\}^l, \text{ the sender picks } b_j' \leftarrow \{0, 1\} \text{ for } j = 1, \ldots, \kappa, \text{ sets } A_w \leftarrow B + \sum_{i=1}^l b_i A_i \in \mathbb{Z}_q^{n \times m} \text{ and } F_w \leftarrow (A_0 | A_w) \in \mathbb{Z}_q^{n \times 2m}. \]

For each \( b_j' \), it computes as follows.
1. Choose a uniformly random $s_j \leftarrow \mathbb{Z}^n_q$ and matrices $R_{ij} \leftarrow \{-1,1\}^{m \times m}$ for $i = 1, \ldots, l$ and set $R_{b_j} \leftarrow \sum_{i=1}^l b_i R_{ij} \in \{-l, \ldots, l\}^{m \times m}$.

2. Choose noise vectors $x_j \leftarrow Z^m_q$ and $y_j \leftarrow Z^m_q$, and set $z_j \leftarrow R_{b_j}^T y_j \in \mathbb{Z}^m_q$.

3. Set $c_{0j} \leftarrow u^T s_j + x_j + b'_j \lfloor \frac{q}{2} \rfloor \in \mathbb{Z}_q$ and $c_{1j} \leftarrow F^T w \cdot s_j + \begin{bmatrix} y_j \\ z_j \end{bmatrix} \in \mathbb{Z}_q^{2m}$.

4. Output searchable ciphertext $s_{wj} = (c_{0j}, c_{1j}, b'_j)$ for $j = 1, \ldots, \kappa$.

tw \leftarrow \text{Trapdoor}(pk, sk, w): On the input of the keys and a keyword $w = (b_i, \ldots, b_l) \in \{1, -1\}^l$, the receiver computes as follows.

1. Let $A_w \leftarrow B + \sum_{i=1}^l b_i A_i \in \mathbb{Z}^{n \times m}_q$ and sample $t_w \in \mathbb{Z}_q^{2m}$ as $t_w \leftarrow \text{SampleLeft}(A_0, A_w, T_{A_0}, u, \sigma)^{19}$.

2. Output the trapdoor as $t_w$.

Given $F_w := (A_0|A_w)$, then $F_w \cdot t_w = u \in \mathbb{Z}_q$, and $t_w$ is distributed as $D_{A_0 u, F_w, \sigma}$.

d \leftarrow \text{Test}(t_w, s_w): Given a trapdoor $td$ for a keyword $w = (b_i, \ldots, b_l) \in \{1, -1\}^l$, and $\kappa$ searchable ciphertexts $s_{wj} = (c_{0j}, c_{1j}, b'_j)$ for $j = 1, \ldots, \kappa$ on keyword $w$, it computes as follows.

1. Set $\nu_j \leftarrow c_{0j} - t_w c_{1j} \in \mathbb{Z}_q$ and check if $|\nu_j - \lfloor \frac{q}{2} \rfloor| < \lfloor \frac{q}{4} \rfloor$, set $\nu_j \leftarrow 1$ and otherwise, $\nu_j \leftarrow 0$.

2. If $\nu_j = b'_j$ holds for all $1 \leq j \leq \kappa$, set $d \leftarrow 1$, else $d \leftarrow 0$.

7.3.2.1 Completeness and Consistency

In this section we show the completeness and consistency of our scheme.

\footnote{As shown in [4], $A_0$ is of rank $n$, with a high probability.}
Lemma 6. Given a public-private key pair \((pk, sk) \leftarrow \text{KeyGen}(\lambda)\), a searchable ciphertext \(s_w \leftarrow \text{PEKS}(pk, w)\) and a trapdoor generated by the receiver \(td \leftarrow \text{Trapdoor}(pk, sk, w)\), the proposed scheme is complete.

Proof. To show the completeness of our scheme for \(sw_j := (c_{0j}, c_{1j}, b'_j)\), the Test algorithm should return 1 when \(\nu_j = b'_j\) where \(\nu_j \leftarrow c_{0j} - tdc_{1j}\) for all \(j = 1, \ldots, \kappa\). To affirm this, we work as follows.

\[
\nu_j = c_{0j} - t_w^\top c_{1j}
\]

\[
= u^\top s_j + x_j + b'_j \left\lfloor \frac{q}{2} \right\rfloor - t_w^\top (F_{id}^\top s_j + \begin{bmatrix} y_j \\ z_j \end{bmatrix})
\]

\[
= u^\top s_j + x_j + b'_j \left\lfloor \frac{q}{2} \right\rfloor - u^\top s_j - t_w^\top \begin{bmatrix} y_j \\ z_j \end{bmatrix}
\]

\[
= b'_j \left\lfloor \frac{q}{2} \right\rfloor + x - t_w^\top \begin{bmatrix} y_j \\ z_j \end{bmatrix}
\]

where \(x - t_w^\top \begin{bmatrix} y_j \\ z_j \end{bmatrix}\) is the error term. Based on Lemma 22 in [4], the error term is bounded by

\[
qu \cdot \sigma \cdot l \cdot m \cdot \alpha \cdot \omega \cdot (\sqrt{\log m}) + O(\sigma m^{3/2})\]

For the system to work correctly, one needs to make sure that:

1. \(\alpha < \left[\sigma \cdot l \cdot m \cdot \omega(\sqrt{\log m})\right]^{-1}\) and \(q = \Omega(\sigma m^{3/2})\),

2. \(m > 6n \log q\) so \(\text{TrapGen}\) can operate,

3. \(\sigma\) is large enough so that \(\text{SampleLeft}\) as defined above, and \(\text{SampleRight}\) (which is similar to \(\text{SampleLeft}\), and is used in the proof of [4]) can operate, i.e., \(\sigma > l \cdot m \cdot \omega(\sqrt{\log m})\)

4. For Regev’s [177] reduction to work, set \(q > 2\sqrt{n}/\alpha\).
To achieve these requirements, we set $q \geq m^{2.5} \cdot \omega(\sqrt{\log n})$, $m = 6n^{1+\delta}$, $\sigma = ml \cdot \omega(\sqrt{\log n})$, $\alpha = [l^2 m^2 \cdot \omega(\sqrt{\log n})]^{-1}$. This is based on the assumption that $\delta$ such that $n^\delta > \lceil \log_2 q \rceil$.

Following the results of Lemma 13 and Lemma 19 in [4], setting the parameters of the scheme as suggested above will ensure the right-keyword consistency of our PEKS scheme with a high probability.

**Theorem 9.** The LWE-PEKS scheme is consistent in the sense of Definition 26.

**Proof.** For the Test algorithm to return 1, all the $\kappa$ bits of $b'_j$ and $\nu_j$ for $1 \leq j \leq \kappa$ should match. This implies that given $A_w$ (obtained from the bit string in the keyword $w$), the SampleLeft algorithm should sample short vectors statistically close (i.e., have negligible statistical distance) to $F_w \leftarrow (A_0 | A_w)$. Therefore, our adversary based consistency comes from the Theorem 3.4 in [70] (and the signing algorithm in [169]) that proves the statistical closeness of $t_w$ that is generated by the SampleLeft on the input of $F_w$. Therefore, for the suggested parameters and based on [70, 169], our LWE-PEKS scheme is consistent.

### 7.3.2.2 Security Analysis

Based on [101], the hardness of lattice problems is measured using the root Hermite factor.

Following Lemma 4, to establish the security of our LWE-PEKS scheme, we need to establish the anonymity property of the underlying IBE scheme. In [4], Agrawal et al. proved the security of their adaptive IBE scheme with a strong privacy property called *indistinguishable from random*, which is a stronger security notion as compared to the anonymity property defined in [2, 35]. In the initial proposal of [4], for the security proof of the adaptive variant of the scheme, there was a
restriction where \( q > Q \) where \( Q \) is the number of queries made by the adversary. This restriction was later lifted in by a more refined analysis in [62]. This implies that based on Lemma 4, the resulting LWE-PEKS scheme is secure in the standard model. Based on [101], the hardness of lattice problems is measured using the root Hermite factor. For a vector \( \mathbf{v} \) in an \( N \)-dimension lattice that is larger than the \( n^{th} \) root of the determinant, the root Hermite factor is computed as \( \gamma = \frac{\|\mathbf{v}\|}{\det(\Lambda_{h,q})^{1/n}} \).

For our LWE-PEKS scheme, we follow the suggested parameters in [75, 141] to achieve \( \approx 192 \)-bit security for message and randomness recovery attack with \( \gamma \approx 1.0042 \).

7.3.3 Secure Channel Requirement

Baek et al. [23] highlighted the requirement of a secure channel for trapdoor transmission between the receiver and the server and proposed the notion of Secure-Channel Free (SCF) PEKS schemes. SCF-PEKS schemes require the server to be initialized with a key pair through which the receiver encrypts the trapdoor before sending it over a public (insecure) channel. Upon receiving the encrypted trapdoor, the server will first decrypt the trapdoor before initiating the Test algorithm. Offline keyword-guessing attack, as introduced by Byun et al. [67], implies the ability of an adversary to find which keyword was used to generate the trapdoor. This inherent issue is due to low-entropy nature of the commonly selected keywords and public availability of the encryption key [58]. Since Byun et al.’s work [67], there have been a number of attempts in proposing schemes that address keyword guessing attacks [95, 121, 126]. However, in all of the proposals, once the trapdoor is revealed to the server, keyword guessing attacks remain a perpetual problem [126]. Jeong et al. [126] showed the trade-off between the security of a PEKS scheme against keyword-guessing attacks and its consistency - by mapping a trapdoor to multiple keywords. For our scheme, we can assume a conventional or even post quantum secure [55] SSL/TLS connection between the receiver and the server. We believe such reliable protocols provide the best means for communicating trapdoors to...
the servers. Establishing a secure line through SSL/TLS could be much more efficient than using any public key encryption as in SCF-PEKS. Since in such protocols, after the handshake protocol, all communications are encrypted using symmetric encryption.

7.3.4 Alternative NTRU-based Constructions

Bellare et al. [36] proposed a new variation of public key encryption with search capability called Efficiently Searchable Encryption (ESE). The idea behind ESE is to store a deterministically computed “tag” along with the ciphertext. To respond to search queries, the server only needs to lookup for a tag in a list of sorted tags. This significantly reduces the search time on the server. For ESE to provide privacy, the keywords need to be selected from a distribution with a high min-entropy. To compensate for privacy in absence of a high min-entropy distribution for keywords, the authors suggested truncating the output of the hash function to increase the probability of collisions. However, this directly affects the consistency of the scheme and shifts the burden of decrypting unrelated responses to the receiver. As compared to PEKS schemes, in ESE schemes, the tag can be computed from both the plaintext and ciphertext. This highly differentiates the applications of these two searchable encryption schemes.

In this paper, we focused on PEKS scheme as it does not have consistency issues or a min-entropy distribution requirement, and fits better for our target real-life applications (as discussed in Section 1). Nevertheless, for the sake of completeness, to extend the advantages of NTRU-based encryption [207] to ESE, we also instantiated an NTRU-based ESE scheme based on the encrypt-with-hash transformation proposed in [36]. We compared it with its counterpart which was instantiated based on El-Gamal encryption. Our implementations of NTRU-based ESE and El-Gamal ESE (developed on elliptic curves) were run on an Intel i7 6700HQ 2.6GHz CPU with 12GB of RAM. We observed that encryption for NTRU-based ESE takes 0.011ms where encryption
Table 7.1: Parameter our schemes and their counterparts

<table>
<thead>
<tr>
<th>Schemes</th>
<th>Public Key (Kb)</th>
<th>Private Key (Kb)</th>
<th>SC^‡ (Kb)</th>
<th>TD† (Kb)</th>
</tr>
</thead>
<tbody>
<tr>
<td>BCOP [52]</td>
<td>0.38</td>
<td>0.19</td>
<td>0.57</td>
<td>0.38</td>
</tr>
<tr>
<td>ZI [214]</td>
<td>0.76</td>
<td>0.19</td>
<td>0.89</td>
<td>0.57</td>
</tr>
<tr>
<td>NTRU-PEKS</td>
<td>27.2</td>
<td>32</td>
<td>52</td>
<td>27</td>
</tr>
<tr>
<td>LWE-PEKS</td>
<td>57,216</td>
<td>14,647,390</td>
<td>186,867</td>
<td>112</td>
</tr>
</tbody>
</table>

All parameters are for $\kappa = 192$.

† SC refers to searchable ciphertext (i.e., the output of the PEKS algorithm).
‡ TD refers to trapdoor (i.e., the output of the Trapdoor algorithm).

Table 7.2: Micro benchmark of our schemes and their counterparts

<table>
<thead>
<tr>
<th>Schemes</th>
<th>PEKS (ms)</th>
<th>Trapdoor (ms)</th>
<th>Test (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>BCOP [52]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\kappa = 80$</td>
<td>6.78</td>
<td>1.26</td>
<td>4.55</td>
</tr>
<tr>
<td>$\kappa = 192$</td>
<td>66.31</td>
<td>4.13</td>
<td>60.75</td>
</tr>
<tr>
<td>ZI [214]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\kappa = 80$</td>
<td>48.87</td>
<td>4.12</td>
<td>12.93</td>
</tr>
<tr>
<td>$\kappa = 192$</td>
<td>301.14</td>
<td>10.61</td>
<td>166.12</td>
</tr>
<tr>
<td>NTRU-PEKS</td>
<td>$\kappa = 80$</td>
<td>1.97</td>
<td>9.71</td>
</tr>
<tr>
<td>$\kappa = 192$</td>
<td>4.44</td>
<td>31.59</td>
<td>3.40</td>
</tr>
<tr>
<td>LWE-PEKS</td>
<td>$\kappa = 80$</td>
<td>2115.12</td>
<td>814.9</td>
</tr>
<tr>
<td>$\kappa = 192$</td>
<td>3727</td>
<td>1509.19</td>
<td>1214</td>
</tr>
</tbody>
</table>

All execution times are reported for one execution.

in El-Gamal ESE takes 2.595ms. As for decryption, NTRU-based ESE takes 0.013ms and El-Gamal ESE takes 0.782ms. The differences are substantial, since the NTRU-base ESE is $236\times$ and $60\times$ faster in encryption and decryption, respectively.

7.4 Performance Analysis and Comparison

We first give the analytical performance comparison and then describe our experimental setup and evaluation metrics. We then provide a detailed performance analysis and cost dissection of our scheme in a cloud setting. To the best of our knowledge, this is the first deployment of PEKS schemes in real-life cloud infrastructure with public data sets.

7.4.1 Analytical Performance Comparison

In this section, we first present a set of recommended parameters for our scheme which were used in our instantiations and then discuss the analytical performance from the computation
7.4.1.1 Parameters

For NTRU-PEKS, with 80-bit security, we set $N = 512$ and $q = 2^{23}$. For 192-bit security, we set $N = 1024$ and $q = 2^{27}$. Following [88], the Gram-Schmidt norm of coefficients of the private key is set as $s \approx \sqrt{\frac{qe}{2}}$, where $e$ is the base of natural logarithm. As illustrated in §7.3.1, the error terms $r, e_1, e_2$ are $N$-dimension vectors with coefficients in $\{-1, 0, 1\}^N$. For LWE-PEKS, with 80-bit security, we set $n = 256$ and $\sigma = 8.35$ and for 192-bit security, we set $n = 512$ and $\sigma = 8$. We set $q$ to be a prime $\approx 2^{43}$. Error terms are sampled from the distribution as defined in Definition 11.

7.4.1.2 Computation

The main performance advantage of NTRU-PEKS stems from the fact that the Test algorithm is executed once on every keyword-file pair for each search query. Therefore, since the Test algorithm of our NTRU-PEKS scheme only requires one convolution product, which is much more efficient than the bilinear pairing operation required in almost all of the existing pairing-based PEKS schemes, we achieve noticeable performance gains in terms of the end-to-end delay. The dominant operations of the PEKS algorithm in our NTRU-PEKS scheme are two convolution products of form $x_1 \ast x_2$. However, since one of the operands has very small coefficients (i.e., $r \in \{-1, 0, 1\}^N$), the convolution products can be computed very efficiently. Specifically, in our case, since $N$ has been selected as a power-of-two integer, the convolution product can be computed in $N \log N$ operations by the Fast Fourier Transform. The PEKS algorithm in BCOP scheme requires one bilinear pairing operation that is more costly than the convolution products. The Trapdoor algorithm in NTRU-PEKS requires a Gaussian Sampling, similar as in [88, 104]. This is the most costly operation in our scheme. This algorithm in BCOP only requires one scalar multiplication and consequently, it is the fastest. For LWE-PEKS, it requires the same type of operations as in NTRU-PEKS, however, mostly due to the
very large dimension of the parameters, it suffers from significant performance loss as compared to its pairing-based counterpart.

7.4.1.3 Storage and Communication

In terms of storage and communication, our schemes are more storage intensive than their pairing-based counterparts. For instance, in our most efficient scheme (i.e., NTRU-PEKS), the sender needs to store the receiver’s public key of size $N \cdot \log_2 q$. Referring to Table 7.1, for 192-bit security, it can be up to $27.2$ Kb. The searchable ciphertext in NTRU-PEKS scheme $3N \log_2(q)$ bits, which corresponds to $52$ Kb. Trapdoor in NTRU-PEKS scheme $2N \log_2(2s \pi)$ bits, where $s$ defines the norm of the Gram-Schmidt coefficient. As depicted in Table 7.1 the size of the searchable ciphertext in NTRU-PEKS is significantly larger than the one in BCOP [52]. However, in §7.4.3, we show that in the deployment of the schemes, even though our NTRU-PEKS scheme incurs a higher communication overhead, this overhead is insignificant when the end-to-end search delay is considered (as compared to the pairing-based counterparts), even with a moderate-speed network.

For LWE-PEKS, the public key is $n \cdot \log_2 q((l + 2) \cdot m + 1)$ bits, where as alluded to in §7.3.2, $m = 6n^{1+\delta}$ and $l$ is the bit length of the keyword. As suggested in [4], $\delta$ should be set such that $n^\delta > \lceil \log_2 q \rceil$. The searchable ciphertext size is $\kappa(2m + 1) \log_2 q$. As depicted, in Table 7.1 the parameter sizes of LWE-PEKS is significantly larger than those of NTRU-PEKS scheme.

7.4.2 Evaluation Metrics and Experimental Setup

7.4.2.1 Evaluation Metrics

We implemented our NTRU-PEKS scheme based on the preliminary implementations in [29, 174]). As aforementioned, PEKS schemes have potential applications in settings with heterogeneous devices. Since in most of these applications, the smaller devices are conceived to be in the role
of a sender, in our experiments, their performance is evaluated in terms of searchable ciphertext generation and sending it to the server. In other applications (e.g., secure e-mail system), commodity hardware may also generate searchable ciphertext and send it to the e-mail server. Thus, their cost is also evaluated on commodity hardware. The receiver/auditor generates a trapdoor to search over the database and process the results. In most applications, the receiver is conceived to be equipped with a commodity hardware (e.g., Laptop). Therefore, we evaluated trapdoor generation and sending it to the server on commodity hardware only.

7.4.2.2 Software Libraries and Hardware Configurations

We fully implemented our NTRU-PEKS scheme in C++, using NTL [190], ZeroMQ and b2 libraries. NTL library was used for low-level arithmetic and matrix operations whereas ZeroMQ was used for network communication. b2 library is a portable C implementation of high-speed Blake2 hash function [21]. Blake2 is used in the full implementation of NTRU-PEKS to map keywords to vec_ZZ type. More specifically, we used Blake2 as a pseudorandom function (PRF), in our Trapdoor and PEKS algorithms. The implementation of the pairing-based counterpart [52] was obtained from MIRACL library, that was provided as a simulation. We extended this implementation for real cloud setting, therefore, in addition to MIRACL, ZeroMQ library is also used in this implementation. Elliptic curves for BCOP scheme were selected based on MIRACL, specifically, we used MNT curve with \( \kappa = 80 \)-bit security (with embedding degree \( k = 6 \)) and KSS curve with \( \kappa = 192 \)-bit security (with embedding degree \( k = 18 \)). We made both of the implementations open-sourced for further improvements and adoption\(^{20}\). For the schemes in the standard model, we only simulated the implementation of our LWE-PEKS and its pairing-based counterpart [214]. Therefore, due to their poor computational performance and parameter sizes (for LWE-PEKS), we decided not to go ahead

\(^{20}\)https://github.com/Rbehnia/Full_PKEKS
with the full-fledged implementation and merely provide an overview of how costly these schemes could be as compared to the schemes in the random oracle model.

As the commodity hardware, we used an Intel Core i7-6700HQ laptop with a 2.6 GHz CPU and 12GB RAM. As a low-power device, we selected ARM Cortex A53 processor, due to its flexibility and low-power consumption [204]. Although it is a low-power device (can run with a small 2200 mAh battery), ARM Cortex A53 is equipped with a 64-bit 1 GHz processor and 1 GB SDRAM. It is extensively preferred in practice since it combines powerful processing power with low energy consumption [204]. At the server-side, we used an Amazon EC2 instance located in Oregon, with a single core Intel(R) Xeon(R) CPU E5-2676 operating at 2.4GH, 2GB RAM and 250 GB SSD. Since our parameter sizes are larger than pairing-based schemes, we preferred to be conservative and selected a moderate-speed network with a 75 Mbps connection for both commodity hardware and ARM processor. The ping to the server is measured as 25.23 ms and 26.78 ms from commodity hardware and ARM processor, respectively. In our experiments, we used subsets of publicly available Enron e-mail dataset.

One can notice that despite the strong security assurances that schemes in the standard model offer, the computation and storage overhead (especially for LWE-PEKS) marks their impracticality, especially with the current state computation capabilities of commodity hardware.

7.4.3 Performance Evaluation

Table 7.2 shows the experimental results obtained from the full-fledged implementation of our lattice-based PEKS schemes and the one proposed in [52, 214]. As illustrated, our NTRU-PEKS scheme is $15 \times$ and $18 \times$ faster than BCOP in the PEKS and Test algorithms, respectively. However, due to costly Gaussian Sampling, the Trapdoor algorithm is approximately $8 \times$ slower than the one in BCOP scheme. We note that due to the latest advancements in Gaussian sampling techniques,
one can instantiate the Test algorithm with more recent algorithms, for instance, using the method proposed in [90]. As for communication, sending one searchable ciphertext (for $\kappa = 192$ bits, equivalent to 52 Kb of data), with our moderate-speed network setting takes only 94 ms which is only 12 ms higher than that of BCOP. This communication cost becomes much costlier in LWE-PEKS. Sending a Trapdoor, which needs to be done once per receiver query on the server, takes 92 ms in our scheme which is only 12 ms more than that in the BCOP scheme.

As for our implementation on ARM processor, the searchable ciphertext generation takes 22.58 ms in our scheme which is $40 \times$ faster than that of BCOP. As aforementioned, we did not benchmark the Trapdoor algorithm since in most applications, these devices operate as senders. The communication of one searchable ciphertext in our scheme takes 100 ms on ARM processor with the aforementioned network setting which is only 14.5 ms higher than that of BCOP scheme.

7.4.4 Detailed Analysis and Cost Dissection

Figure 7.2 and Figure 7.3 show the cost dissection of the end-to-end delays for NTRU-PEKS and LWE-PEKS when the receiver searches over the database for $\kappa = 192$ based on our full-fledged implementation and simulation results, respectively. As depicted, server computation which is
Figure 7.3: End-to-end search time of LWE-PEKS for $\kappa = 192$

running the Test algorithm, dominates the total cost for both of the schemes since it is executed once per each keyword-file pair (linear) in the database. Disk access time on the server is also linear to the number of keyword-file pairs in the database, however, it is much faster as compared to the Test algorithm. Therefore, it contributes to 3% of the total end-to-end delay in NTRU-PEKS. Due to the larger parameter sizes in LWE-PEKS this increases to 15% of the total end-to-end delay.

We conducted experiments with varying sizes of database, up to 200,000 keyword-file pairs. As depicted in Figure 7.4, for 200,000 keyword-file pairs, BCOP algorithm takes 3.34 hours, whereas

Figure 7.4: End-to-end search time comparison for $\kappa = 192$
NTRU-PEKS takes 11.71 minutes. Both of these times are dominated by the Test computation on the server-side, since the receiver generates and sends the Trapdoor only once for each search.

We implemented PEKS on both the commodity hardware and ARM processors. We observed significant improvements on the ARM processor, wherein specifically, for $\kappa = 192$, the PEKS algorithm only takes 22.58 ms which is $40 \times$ faster than BCOP in ARM Cortex A53. This difference is mainly due to the fact that our PEKS algorithm does not require any expensive operations (e.g., exponentiation or pairing computation) and matrix operations can be efficiently computed on a wide range of devices. Another advantage of our scheme is the energy efficiency (longer battery life) on these devices. The energy consumption of a device is linear with the computation time ($E = V \cdot I \cdot t$, where $E =$ energy, $V =$ voltage, $I =$ current and $t =$ time). Therefore, with our NTRU-PEKS scheme, battery replacement cost and cryptographic overhead on energy consumption would potentially decrease significantly.

We observed that although parameter sizes for NTRU-PEKS are much larger than the pairing-based counterpart, they do not significantly affect the communication delay. More specifically, as aforementioned communication difference between NTRU-PEKS and BCOP is only around $10 - 15$ ms. The reason behind this is the round-trip delay time (RTT) from our moderate-speed home network (which is located in the same state as the server, i.e., Oregon) to the server is 25.23 ms and 26.78 ms, for commodity hardware and ARM processor, respectively. With a three-way handshake in TCP, RTT dominates the total communication cost, resulting in an insignificant difference between our NTRU-PEKS and BCOP. This shows that, although NTRU-based schemes have larger parameters, their computational results in a lower end-to-end delay as compared to their communication efficient counterparts (e.g., pairing-based schemes). When the schemes in the standard model are considered, based on our simulated results, as depicted in Figure 7.4, the gap
between the BCOP scheme and ZI scheme (which is secure in the standard model) is much less than
the gap between NTRU-PEKS and the LWE-PEKS which is secure in the standard model.

7.4.5 Discussion

We presented the first full-fledged implementations for PEKS schemes, and make our
implementation open-sourced for further adoption and improvements. Our experiments showed that
(i) Test algorithm dominates the total search time since it runs $O(L)$ times (linear with the number
of keyword-file pairs, $L$). (ii) The efficiency of PEKS algorithm is also crucial since it is to be run on
energy-constrained devices in heterogeneous settings. (iii) Given that lattice-based schemes have
larger parameters and require significantly larger ciphertexts/trapdoors to be transferred, in a real
cloud setting, with a moderate speed network, the communication time difference with pairing-based
schemes could be insignificant.

For real-world cases with large databases, our NTRU-PEKS scheme seems to be the only
practical solution at this moment. We believe that this is one of the main aspects of our scheme
that makes it an attractive candidate to be implemented for real-world applications.

7.5 Related Work

Searchable encryption can be instantiated from both symmetric or asymmetric key settings. Song et al. [191] proposed the first SE scheme that relies on symmetric key cryptography. Kamara et al. [127] proposed the first DSSE scheme to address the limitation of its static ancestors. While being highly efficient, symmetric SE schemes are more suitable for applications that involve a single client who outsources her own data to the cloud relying on her private key.

In this paper, given the target applications that need multiple heterogeneous entities to
create searchable encrypted data, our focus is on SE schemes instanced in asymmetric settings.
In particular, we concentrate on PEKS, as it requires neither specific probability distributions on keywords nor performance/consistency trade-offs as dictated by some other asymmetric alternatives (e.g., ESE as discussed in §7.3.4). In PEKS schemes, decryption and trapdoor generation take place using the private key of the receiver, while any user can use the corresponding public key to generate searchable ciphertext. With a few exceptions, all of the proposed PEKS schemes are developed using costly bilinear pairing operations. The first instance of pairing-free PEKS schemes are constructed by Crescenzo and Saraswat [81] based on the IBE scheme in [76], which is constructed using quadratic residue for a composite modulus. Khader [129] proposed the first instance of such schemes in the standard model based on a k-resilient IBE, she also put forth a scheme which supports multiple-keyword search.

In the Trapdoor algorithm of NTRU-PEKS and LWE-PEKS, following the works in [88] and [4], the receiver uses its secret, which is a short basis of the lattice, as a trapdoor to sample short vectors. Using the short basis of the lattice as a trapdoor to efficiently sample private keys (trapdoors, in the case of PEKS schemes) has been extensively studied in the literature (e.g., [41, 103, 105, 137, 156]).
References


Appendix A: Copyright Permissions

In the following we provide the permissions for the reproduction of the material in Chapters 3, 4, 5 and 7 from the following publishers.

1. IEEE:

   **Author posting of IEEE copyrighted papers**

   The IEEE Publication Services & Products Board has approved revisions to IEEE’s author posting policy (Section 8.1.9 of the PSPB Operations Manual). The revised policy reaffirms the principle that authors are free to post the accepted version of their articles on their personal websites or those of their employers (Authors of IEEE open access articles may freely post the final version of their papers). The policy provides that IEEE will make available to each author a pre-print version of that person’s article that includes the Digital Object Identifier, IEEE’s copyright notice, and a notice showing the article has been accepted for publication. The policy also states that authors are allowed to post versions of their articles on approved third-party servers that are operated by not-for-profit organizations.

   - FAQ on Authors’ Posting of Accepted IEEE Papers (PDF, 210 KB)
   - IEEE Copyright Form
   - Policy on Electronic Dissemination

2. Springer:

   **Author reuse**

   Please check the Copyright Transfer Statement (CTS) or Licence to Publish (LTP) that you have signed with Springer Nature to find further information about the reuse of your content.

   Authors have the right to reuse their article’s Version of Record, in whole or in part, in their own thesis. Additionally, they may reproduce and make available their thesis, including Springer Nature content, as required by their awarding academic institution. Authors must properly cite the published article in their thesis according to current citation standards. Material from: ‘AUTHOR, TITLE, JOURNAL TITLE, published [YEAR], [publisher - as it appears on our copyright page]’

   If you are any doubt about whether your intended re-use is covered, please contact journalpermissions@springernature.com for confirmation.
3. ACM:

**Reuse**

Authors can reuse any portion of their own work in a new work of their own (and no fee is expected) as long as a citation and DOI pointer to the Version of Record in the ACM Digital Library are included.

- Contributing complete papers to any edited collection of reprints for which the author is not the editor, requires permission and usually a republication fee.
- Authors can include partial or complete papers of their own (and no fee is expected) in a dissertation as long as citations and DOI pointers to the Versions of Record in the ACM Digital Library are included. Authors can use any portion of their own work in presentations and in the classroom (and no fee is expected).
- Commercially produced course-packs that are sold to students require permission and possibly a fee.
**About the Author**

Rouzbeh Behnia was a Ph.D. student at Oregon State University (OSU) from 2016 to 2018 and then transferred to the University of South Florida (USF) from 2019 to 2021.

Rouzbeh received his Bachelors and Masters from Multimedia University (MMU), Malaysia. He was an assistant lecturer and then a lecturer at MMU 2013-2016.

His research interests include applied cryptography, cybersecurity in IoT, and trustworthy blockchains.