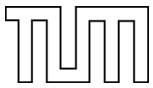

Technische Universität München

Department of Mathematics

Master's Thesis

Privacy and Trust in Adversarial Data Retrieval - Verifiable Computation Techniques for Private Queries on Malicious Servers

Renate Ulla Schwank

Supervisor: Prof. Dr. Violetta Weger

Advisor: Dr. Svenja Lage

Industry Partner: German Aerospace Center (DLR)

Submission Date: 14.03.2026

I assure that this master's thesis is entirely the result of my own work except where otherwise indicated. It is supported only by the declared resources. In particular, AI has solely been used to assist in the correct linguistic formulation and preparation of literature research, in accordance with the official guidelines on AI usage from TUM university [59].

Munich, 14.03.2026 Renate Schwank

Acknowledgments

I want to give thanks to all the people who supported me throughout the writing journey of this thesis. First and foremost, I am grateful to my advisor Svenja Lage for always finding time for discussions on and conversations beyond the topic of this work. Her guidance, advice and attentive ear were of great value to me. Furthermore, I thank my supervisor at TUM, Prof. Violetta Weger, for her openness regarding the topic, her quick and valuable feedback and her general help with all matters regarding the thesis.
Also, I want to mention my entire working group at DLR with our group leader Hannes Bartz. Thank you for all the coffee and cake breaks and for creating a welcoming and open environment in general. I am grateful for the opportunity I was given to work with the DLR on this engaging subject.
Last but not least, I thank all my family and friends for their encouragements and the time they shared with me to get my mind off the thesis and to restore.

Abstract

The task of secure data retrieval from remote entities is subject to ongoing research and of growing relevance in the context of outsourced computation. Cryptographic primitives like Private Information Retrieval (PIR) and Fully Homomorphic Encryption (FHE) are designed to protect a client's privacy when querying databases or delegating expensive computations to external servers. Yet, many schemes assume the semi-honest threat model, where computing parties may only behave in a curious manner but otherwise have to adhere to the specified protocol. Hence, their security guarantees might be insufficient for arbitrary deviations from the protocol. Moreover, malicious parties might deliberately manipulate data to mislead clients into accepting false results. To defend against these attacks on correctness, methods of verifying responses are integrated into existing primitives. Our first main contribution is therefore to provide formal definitions of general blueprints for verifiable PIR (vPIR) and verifiable FHE (vFHE). We also present overviews of existing work to show different techniques of establishing the property of verifiability. The second part of the thesis focuses on the connection between PIR and FHE. It is known that PIR can be constructed from FHE. We examine if verifiability is preserved under one concrete construction. That is, starting from a verifiable FHE scheme, we are interested in whether the PIR scheme derived from it still exhibits this property. As we show, under a certain set of assumptions, vFHE schemes can indeed serve as basis for building vPIR schemes. In particular, vPIR protocols obtained in this way offer strong guarantees for both data privacy and correctness and are therefore well-suited for settings where malicious adversaries may be present.

Kurzfassung

Es ist Gegenstand laufender Forschung, den sicheren Datenabruf von externen Parteien zu gewährleisten. An Bedeutung gewinnt diese Frage besonders im Kontext ausgelagerter Berechnungen. Kryptographische Primitive wie Private Information Retrieval (PIR) und Fully Homomorphic Encryption (FHE) sind darauf ausgelegt, bei der Anfrage von Datenbanken oder der Auslagerung teurer Berechnungsschritte an externe Server die Integrität von Nutzerdaten zu wahren. Viele Verfahren sind jedoch lediglich für das sogenannte semi-honest Bedrohungsmodell konzipiert worden. In diesem dürfen neugierige Teilnehmer zwar versuchen, möglichst viele Informationen von den ausgetauschten Nachrichten abzuleiten, müssen sich jedoch im Grundsatz an das vorgegebene Protokoll halten. Daher sind die Sicherheitsgarantien dieser Verfahren meist ungenügend, wenn jegliche Abweichung vom Protokoll angenommen werden muss. Zudem können bösartige Parteien Daten absichtlich manipulieren, um Nutzer von der Richtigkeit falscher Antworten zu überzeugen. Um gegen diese Angriffe auf die Korrektheit von Daten abzusichern, werden bestehende Primitive um Methoden zur Verifizierung von Ergebnissen erweitert. Einer unserer Hauptbeiträge besteht daher darin, formale Definitionen für allgemeine Protokolle zu verifizierbarem PIR (vPIR) und verifizierbarem FHE (vFHE) aufzusetzen. Darüber hinaus geben wir einen Überblick zu bestehenden Arbeiten auf diesen Gebieten, um verschiedene Methoden zur Umsetzung der Verifizierbarkeit vorzustellen. Im zweiten Teil der Arbeit widmen wir uns der Verbindung zwischen PIR und FHE. Bereits bekannt ist die Möglichkeit, PIR-Protokolle auf Grundlage von FHE zu konstruieren. Wir untersuchen, ob die Eigenschaft der Verifizierbarkeit durch eine solche Konstruktion erhalten bleibt. Genauer betrachten wir, ob die von verifizierbaren FHE-Verfahren erzeugten PIR-Protokolle selbst ebenfalls verifizierbar sind. Wie wir zeigen, gilt dies unter bestimmten Annahmen in der Tat. Insbesondere entstehen auf diese Weise vPIR-Protokolle, die sowohl für die Datensicherheit als auch die Richtigkeit von Ergebnissen starke Garantien versprechen und daher gut geeignet sind, wenn bösartige Teilnehmer toleriert werden müssen.

Contents

List of Figures

List of Tables

1 Introduction

In recent years, concerns about privacy in the digital world have grown. Thereupon, privacy-preserving and -enhancing techniques have gained more attention and have become subject of intensified research. Among these, *Private Information Retrieval (PIR)*, first introduced by [16] in 1995, is a foundational approach to querying remote databases without revealing the identity of the accessed files. Furthermore, this privacy guarantee should introduce only small overhead for the user compared to requesting files in the clear. When asking for perfect privacy in the information-theoretic sense, it was shown in [16] that no single-server scheme, where only one server holds the database and answers the user's queries, does better than the trivial solution of the client downloading the entire database. Therefore, multi-server protocols were proposed that distribute copies of the database to at least two different servers. Clients then privately query all servers and combine the answers to arrive at the file of interest. These schemes, however, typically have one major drawback: They assume the servers to be non-colluding. Since we consider this unrealistic for many real-world applications, we keep to the single-server setting and instead relax the privacy requirement. That is, we only protect against computational adversaries that mount their attacks in polynomial time and have limited computational resources. As a consequence, we view PIR as a two-party protocol between a client issuing file requests and a server hosting the corresponding database.

In early works, both parties were typically considered to be semi-honest, meaning that they may be curious and exploit transmitted messages to learn more about each other's data, yet are not allowed to deviate from the protocol. As already identified in [16], the privacy requirement heavily relies on the semi-honest assumption and usually falls short when exposing the scheme to malicious servers. This argument was strengthened by Kushilevitz et al. [45], who described an attack based on this dependency that many schemes existing up to that point were prone to. Later, this attack became known as *selective-failure attack* [24]. Since maliciousness typically knows no bounds and it is unrealistic that a client can always put its trust in a server, it is vital that there are scheme-inherent protections against fraudulent behavior. More concretely, clients should be shielded from any server-side protocol violations, whether it be breaches of security or deceptions regarding the correctness of answers. In 2008, the necessity was stressed to forward research on securing PIR against dishonest servers [3]. This lead to the proposal of the first *verifiable PIR* scheme in the multi-server setting offering a solution to the issue of privacy and trust in the context of cloud computing [67]. In particular, a verification algorithm is introduced, checking the correctness of the responses obtained from the server. The idea was then transferred to the single-server setting in [61, 70]. This notion of single-server *verifiable PIR* is at the core of interest of this thesis. In verifiable PIR, alongside the usual PIR protocol, verification material is produced and passed between the participating parties. The server is tasked with proving the accordance of its response with the expected protocol outcome. On the client side, the response is only accepted if this proof is found to be convincing. Otherwise, the client aborts the protocol and can decide to reprocess the query or denounce the server of behaving dishonestly. Like for general PIR, both retrieving the desired file and verifying its correctness should be efficient for the client, i.e., cheaper than downloading the full database.

When talking about malicious servers or adversaries, we mean parties that can be modeled

by any fixed Probabilistic Polynomial-Time (PPT) algorithm [46, 34, 51]. In [34], three classes of misbehavior are identified. Firstly, adversaries may refuse to participate in the protocol at all. Secondly, after engaging in the interaction, the protocol might be aborted in mid-execution before the client receives its final output. Lastly, the server can freely alter its initial input or compute any arbitrary function on the input data. Since in the first two cases there is no real harm done to the client, other than having to restart the protocol or trying with a different server, we focus on the third scenario. In the context of PIR, substituting local input data can be interpreted as freely choosing the database used for the PIR computations. This results in arbitrary outputs and undermines any aspirations of reliability [7, 17]. To illustrate the potential threats arising from wrongful answers, we put PIR queries into context and give some concerning examples [17, 62].
Suppose, the client queries a server hosting a database of public keys. It might be tricked into accepting a fraudulent key for which the server knows the corresponding secret key. Therefore, all messages encrypted under the fake public key can effortlessly be decrypted and read by the server. In case the database consists of DNS addresses, a client could either be prevented from accessing certain websites or be redirected to adversarial services. Moreover, when intending to stream a movie or download a software update, a malicious server could replace the data by malware and thereby infect the client's computer.
While the above examples show that unprotected clients are subject to the adversary's will, untruthful answers are not always deliberate. They can also be caused by software bugs, network delays or any kind of failure that emerges during computation [13, 62, 39]. By verifying PIR responses, such errors are detected likewise and can be corrected by rerunning the PIR protocol. This saves the client from processing faulty data that might lead to inconsistencies or failures at later stages of computation or simply falsify the final outcome.

Fully Homomorphic Encryption (FHE) is a cryptographic primitive that enables the processing of encrypted data without access to the underlying information. It is a rather recent field, with C. Gentry being the first to introduce a scheme supporting the homomorphic computation of arbitrary functions in 2009 [31]. Its properties are attractive in a variety of settings [29]. On the one hand, it allows to outsource costly computations from a computationally weak but trusted domain to a more powerful yet possibly malicious party. With the emergence of cloud computing, this outsourcing to unreliable services has become of special concern. On the other hand, computations can be distributed among a large number of entities without compromising the confidentiality of the underlying data. Hence, FHEs are typically deployed in fields concerned with sensitive data, for example the medical or financial sector [43].
Similar to the context of PIR, different assumptions about the honesty of the computing parties can be made. While for PIR the setting of a semi-honest server might be plausible in some cases, FHEs usually already conjecture the computing entity to be somewhat dishonest. Yet, several previous works [60, 15, 50, 64, 43] demonstrate that it would be too shortsighted to rely on FHE alone when entering malicious terrain. Care needs to be taken to account for the full power of malicious adversaries that may deviate from the protocol in any arbitrary way. To be concrete, it is the defining property of homomorphic encryption, namely the malleability of ciphertexts, that exposes a critical point of attack: Since the ciphertext space is closed under operations like addition and multiplication, the

computing party can intentionally alter ciphertexts and exploit the client's reaction to them as a decryption oracle. In the worst case, by repeating this strategy, the server can partially or fully recover the client's secret key used for data decryption. Once the secret key is known, a server can decrypt all stored ciphertexts, not just the ones part of the current computation [43]. These so-called *key-recovery attacks* are therefore an alarming threat to the confidentiality guarantee of FHE schemes [60]. While this and other attacks are detailed in a subsequent chapter, it already illustrates the need for further protection. *Verifiable* FHEs, or short *vFHEs*, answer this request with tools enabling the verification of the responses received. Therefore, verifiability complements the confidentiality guarantees of standard FHE schemes by additionally allowing to audit the correctness of the obtained answers [57]. Only if responses can be certified as being correct, they are decrypted and processed further by the client. Note that in the context of FHEs, it is rather natural to assume maliciousness of the party performing the computations. Since clients hide their local data by encrypting it homomorphically prior to outsourcing, their contents must be sensitive or of high relevance. This gives the computing party an incentive to manipulate the data in some way. What keeps an already distrusted entity from performing operations that are not intended by the user? This is precisely the purpose of vFHE schemes. We remark that malicious behavior is not always aimed at compromising the client's privacy. Since FHE operations are usually expensive, computing services might just try to bypass these heavy computations [43, 71]. Moreover, even if the party entrusted with the computations is indeed semi-honest, it might be compromised by a malicious attacker. Therefore, it again becomes necessary to ask for verification of the responses [60]. Adversarial third parties on the network, however, are of less concern. These can be repelled by moving the communication to secure and authenticated channels that inherently come with separate integrity guarantees [43]. The key difference is that transmission disallows any alteration to the data packages sent, while server-side adversaries may legitimately transform ciphertexts by applying the FHE functionality.

Organization The thesis is structured into three main parts: The section on preliminaries, the formal introduction to verifiable PIR and verifiable FHE and finally, the interplay between these two concepts, manifesting itself in constructing vPIR protocols from vFHE schemes. To prepare for the upcoming definitions and construction, we cover all preliminaries in Section 2. It introduces basic notation and cryptographic concepts which are fundamental to the body of this work. Sections 3 and 4 define the verifiable notions of PIR and FHE by providing a set of algorithms making up the respective schemes. The requirements usually imposed on these algorithms for meaningfulness are also stated. In both cases, variants of the base schemes are outlined together with the new terminology they entail. This emphasizes the depth and the dynamic of the current research on these topics. To move from the general formulation to concrete protocols, we provide non-exhaustive overviews of selected vPIR and vFHE schemes in the form of tables. These also convey intuition for how the desired property of verifiability can be realized. We deliberately present the sections on vPIR and vFHE in a parallel manner to underpin their similarities and motivate the task of converting one into the other. Afterwards, we revisit an existing construction of PIR from FHE in Section 5. To make it more suitable for current vFHE schemes, we reformulate it to support vector and matrix operations and use the first as the basis of our own construction. The last part of this thesis, Section

6, is dedicated to extending the PIR-from-FHE construction of the previous section to account for verifiability. In particular, we show that, under our construction, it is possible to obtain verifiable PIR from verifiable FHE. While we aim at keeping our construction as general as possible, we find it unavoidable to make some assumptions to reason properly about how verifiability can be transferred. These and all further components required to make our conversion work will be mentioned. Section 6 closes with formal proofs that all correctness and security conditions of verifiable FHE are preserved by our construction and hence lead to a meaningful verifiable PIR scheme that withstands attacks under the malicious threat model. Finally, we conclude the thesis in Section 7 with a summary of our main contributions together with an outlook of further research on the conversion from vFHE into vPIR. Concretely, we sketch ideas how to adapt the vPIR-from-vFHE construction for a verification technique not captured in the setup of Section 6. This would extend the applicability of our construction to other existing vFHE schemes.

2 Preliminaries

In this section, we introduce some basic terminology, notation and cryptographic primitives which we use in the later parts of this thesis. For the primitives especially, we opted for concise and comprehensive definitions that extract the core ideas behind each notion. Our purposes do not require knowledge beyond these basic definitions. Yet, of course, we invite the reader to consult further literature for a deepened understanding.

2.1 Basic Notation and Concepts

We first fix some basic notation and notions that are used throughout the thesis. At several points we introduce and refer to specific concepts by means of abbreviations. To maintain an overview, we list all of them in alphabetical order and write out their full name in Table 1. Hereby, whenever a notion is not formally introduced in this work, we also provide an external reference for its precise definition. Otherwise, we point to the respective definition in this document.
Let $[n]$ be the set of integers $\{1, \dots, n\}$ for any $n \in \mathbb{N}$. When writing $\{0,1\}^*$, we mean the set of binary strings of arbitrary finite length. For the usual indicator function, we use the notation $\mathbb{1}$.
Throughout the thesis, we assume the computational rather than the information-theoretic setting. Thus, all algorithms constituting to the definition of cryptographic primitives are subject to a certain degree of uncertainty regarding the compliance with their specification. This degree is bounded by setting the value of a security parameter. We reserve λ as the symbol for this parameter. When defining a variable b, we write $b \coloneqq a$, where a may be a value, another variable or a more complex term. We overload the symbol $|\cdot|$: For real-valued quantities, it is the usual absolute value. Applied to messages or other cryptographic data, it refers to the bit size. Lastly, for circuits or algorithms, it denotes their depths, that is, the number of logical gates they consist of.
A function f is *negligible* if for all constants $c > 0$, there exists a threshold Λ_c such that for all $\lambda \geq \Lambda_c$ it holds that $f < \lambda^{-c}$ [19, 39]. An unspecified negligible function is denoted by $\mathsf{negl}(\lambda)$. Intuitively, $\mathsf{negl}(\lambda)$ vanishes asymptotically faster than the reciprocal of any positive polynomial [70]. When referring to an arbitrary polynomial evaluated at the security parameter, we write $\mathsf{poly}(\lambda)$. Moreover, for a set $\mathcal{S}$ (often $\mathcal{S} = \{0,1\}$), the expression $s \xleftarrow{\$} \mathcal{S}$ describes drawing a sample s from $\mathcal{S}$ uniformly at random.
Let f be a function taking a single value as input. Let $X \coloneqq \{x_1, \dots, x_n\}$ be some finite set of values. Occasionally, we simplify notation by writing $f(X)$ for applying f to every element in X. Formally,

$$f(X) \coloneqq \{f(x_1), \dots, f(x_n)\}.$$

As mentioned before, all adversarial algorithms appearing in the definitions and constructions are assumed to correspond to Probabilistic Polynomial-Time (PPT) Turing machines [46, 34, 51].
To describe the computational as well as the communication complexity of functions, we make use of the big $\mathcal{O}$ notation. Let f and g be two real-valued functions. To express that f asymptotically grows at most as much as the bounding function g, we write

$$f \in \mathcal{O}(g) \iff \exists C > 0, \exists x_0 > 0 \text{ such that } \forall x > x_0 : |f(x)| \leq C \cdot |g(x)|.$$

Table 1: Abbreviation guide.

Notion	Abbreviation	Reference
Algorithm-Based Fault Tolerance	ABFT	[57]
FHE scheme by Brakerski, Fan, Vercauteren	BFV	[26]
FHE scheme by Brakerski, Gentry, Vaikuntanathan	BGV	[11]
non-adaptive Chosen-Ciphertext Attack	CCA1	Def. 2.4
adaptive Chosen-Ciphertext Attack	CCA2	Def. 2.4
Chosen-Ciphertext Verification Attack	CCVA	Def. 4.8
Chosen-Plaintext Attack	CPA	Def. 2.4
Decisional Diffie-Hellman assumption	DDH	[17]
(verifiable) Fully Homomorphic Encryption	(v)FHE	Def. 4.1
Homomorphic Authenticator	HA	[42]
Homomorphic Message Authentication Code	HMAC	[30]
Input-Verifiable Chosen-Ciphertext Attack	IV-CCA	[64]
Locally-Decodable Code	LDC	[25]
Linearly Homomorphic Encryption	LHE	[55]
Linear-Only Homomorphism	LOH	[15]
(Ring) Learning With Errors problem	(R)LWE	[70], [40]
Message Authentication Code	MAC	[5]
Non-Interactive Zero-Knowlege proof	NIZK	[54]
(verifiable) Private Information Retrieval	(v)PIR	Def. 3.1
Probabilistic Polynomial-Time	PPT	Subs. 2.1
Quadratic Arithmetic Program	QAP	[71]
Rank-1 Constraint System	R1CS	[69]
Short Integer Solution problem	SIS	[13]
Succinct Non-interactive ARGument system	SNARG	Subs. 2.2
Succinct Non-interactive ARgument of Knowledge	SNARK	Subs. 2.2
Trusted Execution Environment	TEE	[52]
verified Chosen-Ciphertext Attack	vCCA	[50]
Zero-Knowledge Proof	ZKP	[37]

Some security definitions rely on random variables or distributions to be indistinguishable. Formally, this concept is captured by the following definition.

Definition 2.1 (Computational indistinguishability [35])**.** Let $X = \{X_\lambda\}_{\lambda\in\mathbb{N}}$ and $Y = \{Y_\lambda\}_{\lambda\in\mathbb{N}}$ be two families of random variables. Further, let D be any PPT algorithm that tries to distinguish between the probability ensembles X and Y. That is, it outputs 1 if an input is believed to belong to the ensemble X, and 0 otherwise. Then X and Y are said to be *computationally indistinguishable*, denoted by $X \stackrel{c}{\equiv} Y$, if

$$|Pr[D(X_\lambda) = 1] - Pr[D(Y_\lambda) = 1]| \leq |\mathsf{negl}(\lambda)|$$

for some sufficiently large λ.

2.1.1 Security Games for Encryption Schemes

FHE schemes are special instances of general encryption schemes. At their heart are security guarantees for the ciphertexts produced by their encryption algorithm. Usually, these guarantees are formulated in terms of indistinguishability games. These games capture the idea that ciphertexts should reveal nothing that allows adversaries to infer the underlying secret plaintexts. Hereby, different levels of security are distinguished depending on the attacker's scope. More concretely, during the game, attackers can be granted access to so-called *oracles*, which can be viewed as black boxes that provide a designated service. The levels of security are then a result of varying the type of oracle as well as the stage at which an adversary may query it.
Before stating the general blueprint of indistinguishability games, we first introduce the three main types of oracles that occur in this context. To this end, we expect the reader to be familiar with the basic notion of a general encryption scheme and otherwise recommend [33] or [50] as reference. Now, let $(\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ be a general encryption scheme with key generation function KGen, encryption Enc and decryption Dec.

- *Encryption oracle*: For a given plaintext x, the oracle $\mathcal{O}_{\mathsf{Enc}}$ replies with a corresponding ciphertext $c_x \coloneqq \mathsf{Enc}(x)$.
- *Decryption oracle*: For a given ciphertext c_x, the oracle $\mathcal{O}_{\mathsf{Dec}}$ replies with the corresponding plaintext $x \coloneqq \mathsf{Dec}(c_x)$.
- *Verification oracle*: Assume there to be an additional algorithm Vf that can verify the correctness of a Boolean statement. As an example, assume the statement to be "The input is a valid ciphertext." Then, for a given input c_x, the oracle $\mathcal{O}_{\mathsf{Vf}}$ replies with 1 if c_x satisfies the statement, that is, if c_x belongs to the ciphertext space. Otherwise, the oracle replies with 0.

Note that oracles inherit the probability of error of their underlying algorithms. Therefore, the oracles, too, may output wrong results and decisions with negligible probability.
Let $\mathcal{A}$ be the algorithm of an adversarial party and $\mathcal{O}$ be any oracle not specified further. To indicate that the attacker $\mathcal{A}$ can query oracle $\mathcal{O}$, we use the notation $\mathcal{A}^{\mathcal{O}}$. Access to multiple oracles at the same time is noted as a list of oracles in the superscript.

Remark 2.2. For both, oracles and the big $\mathcal{O}$ notation, we use the symbol $\mathcal{O}$. It should, however, be clear from the context which concept is meant in each case.

Now, we make use of oracles and define the three most common levels of security for general encryption schemes. These are

- Indistinguishability under *Chosen-Plaintext Attacks (IND-CPA)*: The adversary only has access to an encryption but not a decryption or verification oracle.
- Indistinguishability under *non-adaptive Chosen-Ciphertext Attacks (IND-CCA1)*: The adversary has access to an encryption oracle throughout the game. Additionally, it can query a decryption oracle but only before receiving the challenge ciphertext.
- Indistinguishability under *adaptive Chosen-Ciphertext Attacks (IND-CCA2)*: The adversary has access to both, an encryption and decryption oracle throughout the game.

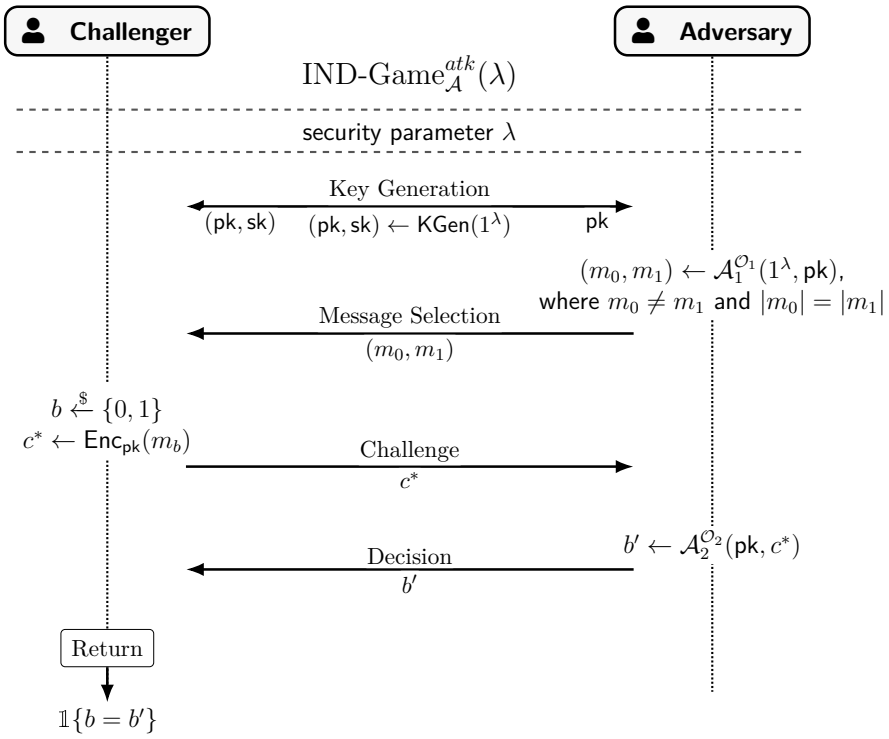

Figure 1: Blueprint for indistinguishability games for public-key encryption schemes.

In the latter part of this thesis, we concentrate on public-key encryption schemes where all parties, including adversaries, can use Enc to encrypt plaintexts. Hence, we define the security notions with respect to this type of encryption. In particular, it becomes redundant to explicitly equip attackers with an encryption oracle, which is why $\mathcal{O}_{\mathsf{Enc}}$ is omitted as superscript in the indistinguishability game.
The basic idea behind the formulation of the game is the following: An attacker $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ may choose any two distinct messages m_0 and m_1 of equal length. Subsequently, an encrypting party, often called the challenger, flips a coin to decide which of the two messages it encrypts. The resulting ciphertext is sent back to the attacker, which is challenged with guessing whether the ciphertext corresponds to m_0 or m_1. Ideally, attackers should be unable to infer anything about the underlying plaintext from a valid encryption. Therefore, it should not be possible to tell m_0 and m_1 apart just from the challenge ciphertext. Hence, in a secure encryption scheme, the attacker guesses correctly with probability $\frac{1}{2}$. The formal definition of the game is given in Figure 1.

Remark 2.3. The formulation of the game shows that it is crucial for the encryption Enc to be probabilistic, instead of deterministic. Otherwise, an attacker can simply encrypt both m_0 and m_1 and compare the results with the challenge ciphertext c^*.

Definition 2.4 (Levels of security [50])**.** Let $\mathcal{E} = (\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ be a public-key encryption scheme and IND-Game$^{atk}_{\mathcal{A}}(\lambda)$ be the indistinguishability game as described in Figure 1. For an adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, define its advantage in the game as

$$\mathsf{Adv}^{atk}_{\mathcal{A}}(\lambda) := 2 \cdot \left| \Pr\left[\text{IND-Game}^{atk}_{\mathcal{A}}(\lambda) = 1\right] - \frac{1}{2} \right|.$$

Then $\mathcal{E}$ has the security level $atk \in \{\text{IND-CPA}, \text{IND-CCA1}, \text{IND-CCA2}\}$, if for any PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ the advantage of $\mathcal{A}$ in the $\text{IND-Game}_{\mathcal{A}}^{atk}(\lambda)$ is negligible, that is

$$\mathsf{Adv}_{\mathcal{A}}^{atk}(\lambda) \leq \mathsf{negl}(\lambda).$$

Hereby, the adversary in Figure 1 is granted access to the oracles

$$\mathcal{O}_1 = \begin{cases} \varepsilon \\ \mathcal{O}_{\mathsf{Dec}} \\ \mathcal{O}_{\mathsf{Dec}} \end{cases} \quad \text{and} \quad \mathcal{O}_2 = \begin{cases} \varepsilon & \text{for } atk = \text{IND-CPA}, \\ \varepsilon & \text{for } atk = \text{IND-CCA1}, \\ \mathcal{O}_{\mathsf{Dec}} & \text{for } atk = \text{IND-CCA2}, \end{cases}$$

where ε is the oracle that always outputs the empty string. In case of IND-CCA2, $\mathcal{A}_2$ is prohibited from asking the oracle to decrypt the challenge c^*.

Remark 2.5. In the above definition, the advantage of winning is normalized to the unit interval $\mathsf{Adv}_{\mathcal{A}}^{atk}(\lambda) \in [0, 1]$. Other works define it to lie in the interval $[0, \frac{1}{2}]$ [50] or $[-1, 1]$ [6]. Which definition one uses, is merely a matter of preference, not meaning.

We expand on different notions of security further, after the formal introduction of verifiable FHE in Subsection 4.1. There, we sketch existing work on scheme constructions that implement these different security levels and put verification oracles more into focus.

2.2 Building Blocks

Our construction of verifiable PIR from verifiable FHE is based on and includes some components that exist as standalone primitives in the field of cryptography. First, verification makes use of short arguments, so-called *Succinct Non-interactive ARGuments (SNARGs)*, testifying to the correct execution of computations. Verifying these arguments is much faster and less computationally expensive than redoing the entire computation oneself to check the conformance of results. Second, it will become essential to bind entities to some of their internal knowledge. *Commitment schemes* can provide this service. We provide the formal definitions of both concepts.

2.2.1 Succinct Non-interactive Arguments

Let $\mathcal{R} \subseteq \{0,1\}^* \times \{0,1\}^*$ be a binary relation consisting of *statement-witness pairs* (x, w) such that the membership of tuples to $\mathcal{R}$ is decidable in polynomial time. Moreover, the size of the witness w should at most be polynomial in the size of the statement x. Such a relation is called $\mathcal{NP}$ *relation*. Its associated language

$$\mathcal{L} \coloneqq \{x \mid \exists w \text{ such that } (x, w) \in \mathcal{R}\}$$

consists of all statements appearing in the relation $\mathcal{R}$ [36].

Example 2.6. To illustrate this rather abstract definition, we provide the instructive example of the decisional integer factorization problem [18]. Let $n, k \in \mathbb{N}$ be two integers. The question is, whether there exists some integer $d > 1$ that is strictly smaller than k and that divides n. This decision problem translates to the $\mathcal{NP}$ relation

$$\mathcal{R}_{fact} \coloneqq \{((n, k), d) \mid n, k, d \in \mathbb{N} \text{ with } 1 < d < k \text{ and } d | n\}$$

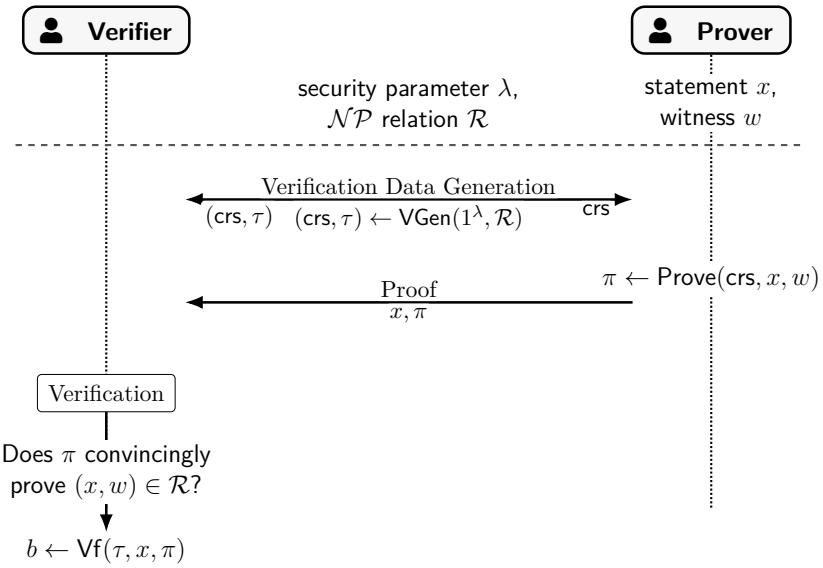

Figure 2: Interactive protocol for SNARG proof systems.

with its language

$$\mathcal{L}_{fact} \coloneqq \{(n, k) \mid \exists d \in \mathbb{N} \text{ such that } 1 < d < k \text{ and } d|n\}.$$

Given a statement (n, k) together with an alleged witness d, checking the conditions $1 < d < k$ and $d|n$ can easily be done in time linear in n.

Before presenting a formal definition of succinct non-interactive argument systems, we take the example of the $\mathcal{NP}$ relation $\mathcal{R}_{fact}$ and demonstrate the service SNARGs offer in this context. For given parameters n and k, the task in question is to prove to a client that one knows a value d which meets the constraints $1 < d < k$ and $d|n$. Crucially, the proof should not reveal the value of d directly but, at the same time, it has to be fully convincing. Moreover, the proof should be short and verifying it should need little computational effort. A security parameter specifies how hard it is to forge such proofs.

Let us proceed with defining SNARGs formally. A visual representation of the protocol is provided in Figure 2.

Definition 2.7 (Succinct Non-Interactive Argument System - SNARG [32, 8, 10, 39])**.** Let λ be some suitable security parameter and $\mathcal{R}$ be an $\mathcal{NP}$ relation decidable in $\mathsf{poly}(\lambda)$ time. Then, a *Succinct Non-interactive ARGument system (SNARG)* for $\mathcal{R}$ is a triple of PPT algorithms $(\mathsf{VGen}, \mathsf{Prove}, \mathsf{Vf})$ defined by

- $\mathsf{VGen}(1^\lambda, \mathcal{R}) \to (\mathsf{crs}, \tau)$: For verification, a *common reference string* crs and *verification state* τ is *generated* according to the security parameter λ and the relation $\mathcal{R}$. While crs is published, τ is only shared with the verifier.

- $\mathsf{Prove}(\mathsf{crs}, x, w) \to \pi$: The *prover* takes the common reference string crs, a chosen *statement* x and a *witness* w as input and outputs an argument π proving $(x, w) \in \mathcal{R}$.

- $\mathsf{Vf}(\tau, x, \pi) \to b$: Based on the verification state τ and the statement x, the *verifier* outputs a bit b. If the proof π is accepted, $b = 1$; otherwise, $b = 0$.

Remark 2.8. To keep τ hidden from the prover, VGen must either be performed by the verifier itself or by an external trusted party.
The term *non-interactive* indicates that no additional messages are sent before the verifier decides on its response to the statement x and proof π.

SNARGs are subject to three requirements: Completeness, soundness and succinctness as defined below. We remark that in all three, the security parameter λ is the same and determined by the relation $\mathcal{R}$.

Definition 2.9 (Completeness for SNARGs [32])**.** Intuitively, SNARGs are *complete* if all honestly computed proofs for actual elements of $\mathcal{R}$ are accepted by the verifier with high probability. Formally, assuming $(x, w) \in \mathcal{R}$ and an honest prover, we have

$$\Pr\left[\mathsf{Vf}(\tau, x, \pi) = 0 \;\middle|\; \begin{array}{r}(\mathsf{crs}, \tau) \leftarrow \mathsf{VGen}(1^\lambda, \mathcal{R}) \\ \pi \leftarrow \mathsf{Prove}(\mathsf{crs}, x, w)\end{array}\right] \leq \mathsf{negl}(\lambda).$$

Definition 2.10 (Soundness (adaptive) for SNARGs [32, 8])**.** A SNARG is said to be *(adaptively) sound* if it is unlikely for an attacker to produce a successfully verifying proof for a statement that is not part of the language. Concretely, let $\mathcal{A}$ be a PPT algorithm imitating the prover's part. Then, for an (adaptively) sound SNARG for $\mathcal{R}$ we require

$$\Pr\left[\begin{array}{c}\mathsf{Vf}(\tau, x, \pi) = 1 \\ \wedge\, x \notin \mathcal{L}\end{array} \;\middle|\; \begin{array}{r}(\mathsf{crs}, \tau) \leftarrow \mathsf{VGen}(1^\lambda, \mathcal{R}) \\ (x, \pi) \leftarrow \mathcal{A}(\mathsf{crs})\end{array}\right] \leq \mathsf{negl}(\lambda).$$

Remark 2.11. *Adaptive* means that the adversary $\mathcal{A}$ is allowed to choose the statement x after seeing the common reference string. If the statement is fixed beforehand, the soundness is *non-adaptive* [8]. This flavor of soundness, however, is not of relevance for our use case.

To render SNARG-based verification communication-efficient and feasible for low-power parties, one typically constrains the size of the proof π. Depending on the concrete reference, bounds may vary [39, 32, 8]. We opted for one that seems reasonable and introduces no new notation.

Definition 2.12 (Succinctness for SNARGs [39])**.** Intuitively, the *succinctness* property guarantees that the size of the proof π is bounded by a fixed polynomial, which is independent of the size of the statement-witness pair [32]. That is, for $(x, w) \in \mathcal{R}$ and π a proof as generated in the completeness requirement, a SNARG for $\mathcal{R}$ is *succinct* if there is a polynomial p such that the length of π is bounded above by

$$|\pi| \leq p(\lambda, \log(|x| + |w|)).$$

For completeness, we also introduce *SNARKs* – a strengthening of SNARGs which requires a stricter soundness guarantee. *SNARK* stands for *Succinct Non-interactive Argument of Knowledge*, whereby the word "knowledge" indicates that it is equipped with a function that can recover witnesses from given statement-proof tuples, provided that these pass

verification. This additional function is commonly called the *knowledge extractor*. More concretely, SNARKs have the same structure as SNARGs except that they fulfill the subsequent requirement of knowledge soundness instead of mere soundness as stated in Definition 2.10.

Definition 2.13 (Knowledge soundness (adaptive) for SNARKs [8, 60])**.** A SNARK for $\mathcal{R}$ fulfills *(adaptive) knowledge soundness* if the following holds. For every adversarial PPT algorithm $\mathcal{A}$ imitating the prover's part, there is a PPT knowledge extractor algorithm $\mathcal{E}_{\mathcal{A}}$ such that for every auxiliary input $z \in \{0,1\}^{\mathsf{poly}(\lambda)}$ we have

$$\Pr\left[\begin{array}{l} \mathsf{Vf}(\tau, x, \pi) = 1 \\ \wedge\, (x, w) \notin \mathcal{R} \end{array} \middle| \begin{array}{r} (\mathsf{crs}, \tau) \leftarrow \mathsf{VGen}(1^\lambda, \mathcal{R}) \\ (x, \pi) \leftarrow \mathcal{A}(\mathsf{crs}, z) \\ w \leftarrow \mathcal{E}_{\mathcal{A}}(\mathsf{crs}, z) \end{array}\right] \leq \mathsf{negl}(\lambda).$$

Remark 2.14. Sometimes, the notation $(\mathcal{A}|\mathcal{E}_{\mathcal{A}})$ is used to combine the two algorithms $\mathcal{A}$ and $\mathcal{E}_{\mathcal{A}}$ [60, 39]. This emphasizes the close relationship between the attacker and the witness extractor: The extractor algorithm is tailored to the attacker and needs to be granted access to its full internal state. Moreover, for the extractor to be able to produce some knowledge w, the statement-proof pair constructed by the adversary has to pass verification successfully [60].

Example 2.15. We exemplify the strengthened guarantee of knowledge soundness for SNARK proofs by drawing once more on Example 2.6 of integer factorization. In this context, adaptive knowledge soundness ensures the existence of an extractor algorithm $\mathcal{E}_{\mathcal{A}}$ that can efficiently produce an integer d meeting the constraints $1 < d < k$ and $d|n$ for $k, n \in \mathbb{N}$. For this value d, the requirement then bounds the probability of the event that verification passes for the statement $x \coloneqq (n, k)$ and some proof π but $((n, k), d) \notin \mathcal{R}_{fact}$.

Both SNARGs and SNARKs come with certain apprehensions, as they often rely on the random oracle model or (non-)falsifiable assumptions [8, 64]. Hereby, the non-falsifiable assumptions are commonly used for SNARKs [64], while SNARGs can be constructed from the falsifiable counterparts [25]. Since our work only requires the existence of witnesses and not their concrete form, we therefore prefer using the more lightweight SNARGs.
We remark that even for minimal examples, in-depth considerations are required to come up with concrete values for SNARG or SNARK proofs that fulfill all conditions stated above. To get an idea of how to derive such values, we refer the reader to the well-written article by Vitalik Buterin [12]. It provides a step-by-step explanation of how to construct a SNARG proof for knowing the solution to the equation $x^3 + x + 5 = 35$.

2.2.2 Commitment Schemes

Commitment schemes are common cryptographic protocols between two parties, a committer (or sender) and a verifier (or receiver). They serve a two-fold purpose: Assume the committer holds some private value m. Then, in the *commit phase*, the committer produces an output that binds itself to m while keeping it secret at the same time. The binding, or *commitment*, ensures that in a subsequent *reveal phase*, the verifier only accepts if the opening of the commitment is indeed the value m [37].

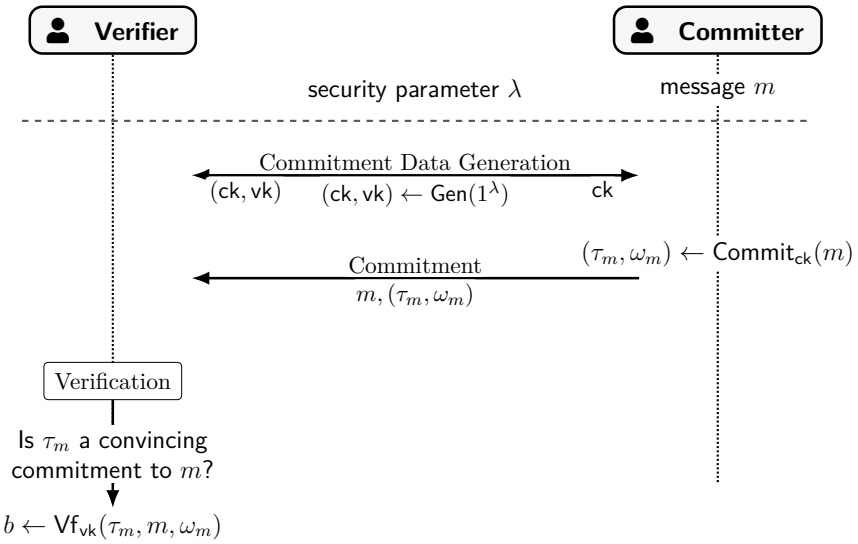

Figure 3: Interactive protocol for commitment schemes.

A physical analogue of commitment schemes is a sealed envelope: The sender binds itself to a message by placing it within the envelope and sealing it tightly. This way, the message remains hidden until someone breaks the seal officially [41, 37].

Definition 2.16 (Commitment scheme [41, 10, 20, 25])**.** A *commitment scheme* Com is a tuple of algorithms $\mathsf{Com} = (\mathsf{Gen}, \mathsf{Commit}, \mathsf{Vf})$ defined by

- $\mathsf{Gen}(1^\lambda) \to (\mathsf{ck}, \mathsf{vk})$: Taking the security parameter λ as input, the *generation* algorithm produces a public key ck for the committer and a private key vk for the verifier.

- $\mathsf{Commit}_{\mathsf{ck}}(m) \to (\tau_m, \omega_m)$: Using the key ck, the committer *commits* to a message m by sending τ_m to the verifier. Additionally, it sends ω_m – material to *open* the commitment and to reveal the underlying message later on.

- $\mathsf{Vf}_{\mathsf{vk}}(\tau_m, m, \omega_m) \to b$: To *verify* the correctness of the message m, the verifier makes use of the verification key vk, the issued commitment τ_m as well as the opening information ω_m. It outputs $b = 1$ if it accepts m as the message committed to and $b = 0$ otherwise.

The protocol interaction is presented in Figure 3. For these algorithms, one typically defines the following set of conditions, asking for correctness, hiding and binding.

Definition 2.17 (Correctness for commitments [10, 25])**.** Assume all algorithms to be executed honestly. Then, a commitment scheme is *correct* if the verifier rightfully accepts the message committed to at the end of the reveal phase. Formally, let m be any message in the message space. Then, we require that

$$\Pr\left[\mathsf{Vf}_{\mathsf{vk}}(\tau_m, m, \omega_m) = 0 \;\middle|\; \begin{array}{r}(\mathsf{ck}, \mathsf{vk}) \leftarrow \mathsf{Gen}(1^\lambda)\\ (\tau_m, \omega_m) \leftarrow \mathsf{Commit}_{\mathsf{ck}}(m)\end{array}\right] \leq \mathsf{negl}(\lambda).$$

Definition 2.18 (Hiding for commitments [10, 20])**.** A commitment scheme is hiding, if it is infeasible to infer the underlying message from the commitment. Formally, let ck be a commitment key, m_0, m_1 be two messages in the message space and $\mathcal{A}$ any PPT adversary. Then, we ask for the following distributions to be computationally indistinguishable

$$\mathsf{Commit}_{\mathsf{ck}}(m_0) \stackrel{c}{\equiv} \mathsf{Commit}_{\mathsf{ck}}(m_1).$$

Remark 2.19. The definition of hiding assumes the commitment algorithm to be probabilistic such that its output is a random variable. This is required for the definition of computational indistinguishability to apply, which is defined over families of random variables. For our own application, however, we mostly concentrate on *deterministic* commitment schemes without hiding. Hereby, the Commit algorithm is a deterministic function; in particular, there is only one fixed commitment for every message. In our case, commitments are made to databases that are public knowledge and therefore they need not be kept secret. Their deterministic nature allows us to test if two commitments stem from the same message by simply checking equality.

Definition 2.20 (Binding for commitments [10, 25])**.** A commitment scheme is *binding* if it is unlikely that more than one message-opening information pair can correspond to a published commitment. That is, a commitment uniquely identifies the underlying message. Formally, let $\mathcal{A}$ be any PPT adversary. Then,

$$\Pr\left[\begin{array}{c} m_0 \neq m_1 \\ \wedge\, \mathsf{Vf}_{\mathsf{vk}}(\tau_m, m_0, \omega_{m_0}) = 1 \\ \wedge\, \mathsf{Vf}_{\mathsf{vk}}(\tau_m, m_1, \omega_{m_1}) = 1 \end{array} \middle| \begin{array}{r} (\mathsf{ck}, \mathsf{vk}) \leftarrow \mathsf{Gen}(1^\lambda) \\ (\tau_m, m_0, \omega_{m_0}, m_1, \omega_{m_1}) \leftarrow \mathcal{A}(\mathsf{ck}) \end{array}\right] \leq \mathsf{negl}(\lambda).$$

Remark 2.21. In the context of PIR, we will most commonly speak of database *digests* instead of database commitments (see e.g. [13, 62]).

Example 2.22. Like for SNARG systems, we provide the example of a simple commitment scheme to illustrate their functionality. To this end, we slightly adapt the above setting of the decisional integer factorization problem. Suppose that a client, in the role of the verifier, determines an integer n. The second parameter k, however, is now provided by a server or prover. As before, the prover wants to demonstrate that it knows an integer d such that $1 < d < k$ and $d|n$. To prevent the server from changing the upper bound on d during the proof phase, a commitment to k is shared with the client initially. Concretely, the interaction between the client and the server takes the following form.

- Let H be a public hash function. The server generates some secret value x, which bit size depends on the security parameter λ. The hash $h := H(k||x)$ is sent to the client, where $||$ denotes the usual concatenation of bits.
- The client stores h locally, saving it for the verification to come.
- For the SNARG proof, the server proceeds as before. That is, for some $1 < d < k$ with $d|n$, it computes a proof π. The response consists of the proof π, the statement (n, k) and the secret x.
- The client first checks the commitment by recomputing $H(k||x)$. If the result equals h, it continues with verifying the proof π. Otherwise, it rejects the response.

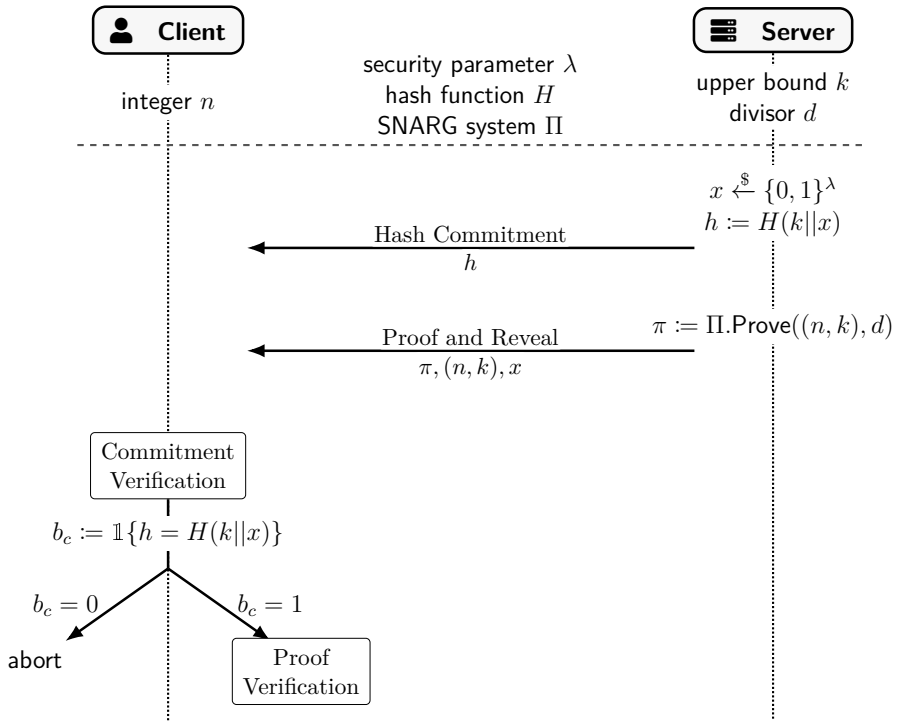

Figure 4: Example of a hash commitment for the decisional integer factorization problem.

This example is captured in Figure 4. We remark that two properties of hash functions are essential to this type of commitment. First of all, preimage resistance ensures that it is computationally infeasible to find some input y such that $H(y) = H(k||x)$. This establishes the hiding property, i.e., the commitment itself does not reveal the underlying message. Secondly, due to collision resistance, it is highly improbable to generate two value pairs (k, x) and (k', x') with the same hash value $h = H(k||x) = H(k'||x')$. Therefore, the commitment h truly binds the server to using k in the subsequent computations.

Due to the generality of our definition, it cannot capture all nuanced properties that come with variants of commitment schemes. Most noticeable, it is not always necessary to have knowledge of the entire ground truth message m in order to verify the commitment. On the contrary, less data might suffice as input to the verification algorithm Vf. This could be data used to compute the commitment in the first place ([13], [55]) or portions of the original message in case only parts of it should be revealed (see *vector commitment schemes* as in [25]). In our upcoming construction of vPIR, we utilize this property to ensure that the client does not need the entire database to verify the correctness of the digest. More concretely, we assume there to be some data Γ that can be used to compute and verify commitments alike. Since this data has to be distributed to all participating parties, we simply model it as part of the commitment and the verification key. This means, when writing ck and vk, we implicitly assume Γ to be contained in the keys.

3 Verifiable PIR

After having set up all required foundational concepts in the previous section, we now come to the first of our two main privacy-preserving cryptographic primitives: Private Information Retrieval (PIR). This section starts with formally introducing verifiable PIR by defining it in terms of the algorithms that make up such protocols and by detailing the properties expected for meaningfulness. It then continues with an outline of the dynamic evolution of vPIR, mentioning interesting nuances that arose and were explored in this course. This is followed by a tabular overview of existing vPIR schemes that picks out some decisive features and allows for a direct comparison of the protocols.

3.1 Defining General vPIRs – Algorithms and Requirements

To prepare for a rigorous definition of vPIR, we first recall the motivation behind standard PIR. In their most general form, PIR schemes enable computationally weak clients to request specific database files from databases hosted by powerful servers in a private manner. As highlighted previously, depending on the trustworthiness of the server and the sensitivity of the data processed, it is essential to defend the clients against deliberate attacks and also accidental response errors. This is reflected in imposing strong requirements on PIR and especially verifiable PIR. The conditions on the latter extend the guarantees of standard PIR schemes. While correctness applies unchanged, the privacy requirement for semi-honest PIR schemes is strengthened and leads to the notion of malicious privacy. Moreover, vPIR schemes have to provide *integrity* - an assurance to be deceived by incorrect answers with only negligible probability.
These additional requirements are mirrored in the algorithms making up vPIR protocols. For standard PIR with semi-honest parties, it suffices to define four algorithms: First, a setup algorithm that generates public parameters and initial states. Second, a query algorithm that produces a query for the file index of interest. Third, a response algorithm that computes the server's answer to the posed query and finally an extraction algorithm that decrypts the response to obtain the desired plaintext file. On top of these, vPIR schemes typically also require a commitment and a verification procedure. The former binds the server to a specific database for the vPIR computations by initially asking for a digest thereof. The latter then verifies the server's response against this previously published digest. If verification fails, the protocol ends with an abort ($\perp$). Otherwise, the answer gets decrypted. Because of this interdependence between the verification and the decryption algorithm, the two are oftentimes combined into one functionality.
For a precise notation, we first describe an abstract blueprint for verifiable PIR. We omit details since the concrete description and naming of the algorithms and requirements depends on the particular scheme in use. The definition of our general vPIR framework arises as a synergy of several existing schemes, most notably [39, 25, 62] and [13]. For this and all other protocol definitions to come, we make one general assumption. Namely, at every stage of the protocol, all parties are assumed to have unlimited access to all public data, their individual secret data, and all parameters they have received so far during the protocol execution. This avoids tracking states and hence streamlines the notation.

Definition 3.1 (Verifiable Private Information Retrieval - vPIR)**.** Let λ be the security parameter, DB be some database of length n. A *verifiable Private Information Retrieval*

scheme, or short *vPIR*, is made up of five algorithms Setup, Digest, Query, Answer and $\mathsf{Decryption}$, which are defined as follows.

- $\mathsf{Setup}(1^\lambda) \to (\mathsf{pp}, \mathsf{sp})$: The setup algorithm takes the security parameter λ as input and outputs public and possibly secret parameters $(\mathsf{pp}, \mathsf{sp})$.
- $\mathsf{Digest}(\mathsf{pp}, DB) \to d$: The server commits to the database DB by computing a digest d thereof and publishing it.
- $\mathsf{Query}(\mathsf{pp}, i) \to q$: The client wants to retrieve the i^{th} entry of the database for some $i \in [n]$. Using public parameters, the query algorithm produces a suitable query q for index i.
- $\mathsf{Answer}(\mathsf{pp}, DB, q) \to a$: For the query q and database DB, the server computes an answer a that is sent back to the client.
- $\mathsf{Decryption}(d, a) \to out$: The answer a is verified by the client utilizing the database digest d. If it is accepted, it gets decrypted and results in some outcome *out*. Otherwise, the answer is rejected and *out* is set to $\perp$.

Remark 3.2. Depending on the scheme design, generating secret parameters during the setup phase might not be necessary (see e.g. [23, 39, 58]). Other works, however, rely on secret material to establish privacy or integrity (see e.g. [66, 62, 70]). In the latter case, the setup algorithm has to be executed by a trusted party.

For a more intuitive understanding of the general vPIR protocol, we illustrate the interaction between two participants, a client and a server, in Figure 5.
With all algorithms in place, we now proceed to defining the conditions on correctness and security.

Definition 3.3 (Correctness for vPIR [25])**.** Let DB be any database of length n and $i \in [n]$ be any index. Assuming an honest server, a vPIR scheme is *correct*, if, with high probability, a query for index i is answered with the i^{th} database entry DB_i. That is,

$$\Pr\left[out \neq DB_i \;\middle|\; \begin{array}{r} (\mathsf{pp}, \mathsf{sp}) \leftarrow \mathsf{Setup}(1^\lambda) \\ d \leftarrow \mathsf{Digest}(\mathsf{pp}, DB) \\ q \leftarrow \mathsf{Query}(\mathsf{pp}, i) \\ a \leftarrow \mathsf{Answer}(\mathsf{pp}, DB, q) \\ out \leftarrow \mathsf{Decryption}(d, a) \end{array} \right] \leq \mathsf{negl}(\lambda),$$

where all vPIR algorithms are executed honestly.

To capture the need to disclose manipulated responses from the server, verifiable PIR schemes have to provide *integrity*.

Definition 3.4 (Integrity for vPIR [25, 39])**.** Intuitively, a vPIR scheme ensures *integrity*, if the client can detect fraudulent responses with high probability. That is, with high probability, whenever an answer passes the client's verification, it is correct and if the answer was manipulated, it is correctly rejected. Formally, let $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ denote

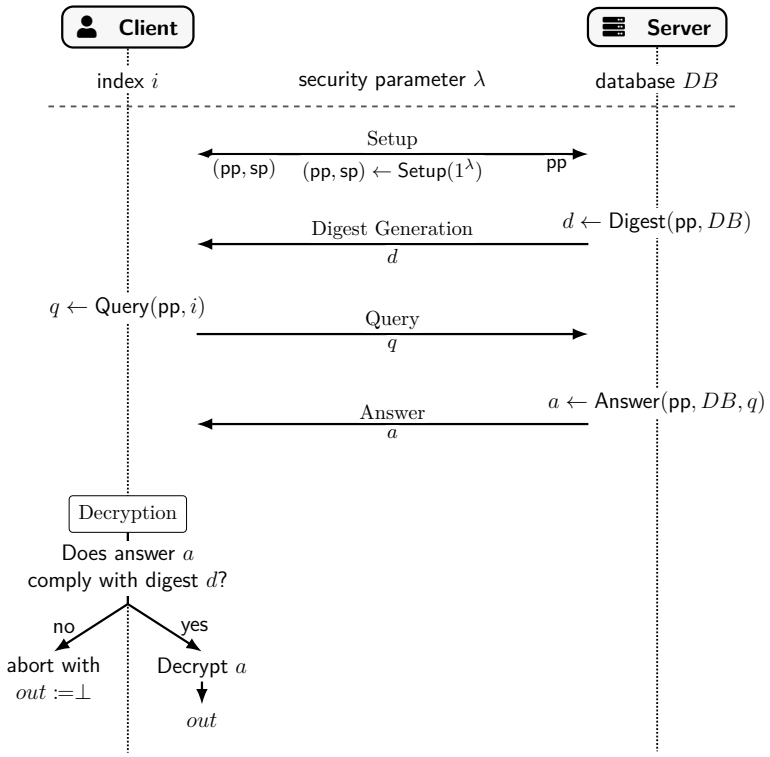

Figure 5: Interactive protocol for the general vPIR blueprint.

an efficient adversary with PPT algorithms $\mathcal{A}_1, \mathcal{A}_2$, emulating the vPIR digest and answer generation, respectively. Then, given some honestly generated public parameters $(\mathsf{pp}, \mathsf{sp}) \leftarrow \mathsf{Setup}(1^\lambda)$, for every adversary $\mathcal{A}$ and possibly malicious digest $d \leftarrow \mathcal{A}_1(\mathsf{pp})$ we require,

$$\Pr\left[\begin{array}{c|c} \exists \text{ a database } DB & q \leftarrow \mathsf{Query}(\mathsf{pp}, i) \\ \text{such that } out \notin \{DB_i, \perp\}, & a \leftarrow \mathcal{A}_2(\mathsf{pp}, q) \\ \text{for any index } i. & out \leftarrow \mathsf{Decryption}(d, a) \end{array}\right] \leq \mathsf{negl}(\lambda).$$

Remark 3.5. We clarify the scope of the integrity definition by addressing two points.

- **Underlying database** In the above definition, the adversary can freely choose the digest d. Crucially, it might not belong to the database the client expects and wants to query. What integrity guarantees in this case is that the client recognizes wrongful behavior with respect to this other, possibly unknown database with digest d. Note, that it is out of the scope of the definition to ensure that *out* relates to the actual database. It merely makes sure that once the server issues the digest of some database, it is bound to using this specific database in all subsequent computations.

- **Inconsistent views** Integrity also prevents the issue of inconsistent views: Consider a multi-round protocol, where several queries are sent after one initial setup and digest generation phase. If not inhibited, a malicious server could, for the same query, give different vPIR answers to different users [25]. This is problematic if a protocol relies on the participating parties having consistent views, for example for majority voting. Integrity imposes that it is improbable to obtain an output that

is accepted but does not agree with the underlying database. Therefore, with high probability, no two different, yet seemingly correct responses to a single index query can exist, ruling out the possibility of inconsistent views.

When verification of the server's response fails, clients react with aborts. If no further thought is put into the scheme, this behavior poses a new point of attack: Malicious servers might only change a single or carefully selective number of database entries. Then, when a client aborts, it can deduce that the queried index corresponds to these altered entries [45]. This *selective-failure attack* therefore violates the basic privacy requirement on PIR schemes, that no information on the requested file index should be revealed. Hence, the basic privacy guarantee needs to be adapted [17]. In [25], this extended privacy notion is called *malicious privacy*. We adopt their terminology.

Definition 3.6 (Malicious privacy for vPIR [25, 23])**.** A vPIR scheme is *maliciously private* if no information about the index queried by the client is leaked. More concretely, the queries for any two indices are indistinguishable and aborts allow no inference about the index of interest. Formally, let $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3)$ denote an efficient adversary consisting of PPT algorithms that emulate the digest ($\mathcal{A}_1$) and answer generation ($\mathcal{A}_2$) and set the value of a bit b ($\mathcal{A}_3$) depending on the occurrence of aborts. Then, a maliciously private vPIR scheme requires that for every adversary $\mathcal{A}$, every fixed database DB of length n and every query index $i \in [n]$, there exists an efficient simulating PPT algorithm Sim such that the following two distributions are indistinguishable

$$\left\{ b \;\middle|\; \begin{aligned} (\mathsf{pp}, \mathsf{sp}) &\leftarrow \mathsf{Setup}(1^\lambda) \\ d &\leftarrow \mathcal{A}_1(\mathsf{pp}, DB) \\ q &\leftarrow \mathsf{Query}(\mathsf{pp}, i) \\ a &\leftarrow \mathcal{A}_2(\mathsf{pp}, d, q) \\ out &\leftarrow \mathsf{Decryption}(d, a) \\ abort &:= \mathbb{1}\{out = \perp\} \\ b &\leftarrow \mathcal{A}_3(abort) \end{aligned} \right\} \overset{c}{\equiv} \left\{ b \;\middle|\; \begin{aligned} (\mathsf{pp}, \mathsf{sp}) &\leftarrow \mathsf{Sim.Setup}(1^\lambda) \\ d &\leftarrow \mathcal{A}_1(\mathsf{pp}, DB) \\ q &\leftarrow \mathsf{Sim.Query}(\mathsf{pp}) \\ a &\leftarrow \mathcal{A}_2(\mathsf{pp}, d, q) \\ out &\leftarrow \mathsf{Sim.Decryption}(d, a) \\ abort &:= \mathbb{1}\{out = \perp\} \\ b &\leftarrow \mathcal{A}_3(abort) \end{aligned} \right\}.$$

Remark 3.7. Slight differences in the above two distributions are crucial. Firstly, on the right-hand side, the setup and query algorithm are simulated by Sim. In particular, the simulated query generation is independent of the entry location i. Hence, computational indistinguishability yields, that the adversary $\mathcal{A}$ cannot have had access to (partial) information about i. Secondly, the decryption algorithm, too, is merely simulated. Therefore, the simulator controls the indicator bit *abort*, which in turn influences the final output b. Thus, the adversary gains no advantage from knowing about aborts. This discussion shows, that the definition indeed captures the ideas of malicious privacy.

The algorithms and notions defined above culminate in our understanding of verifiable PIR. To avoid confusion with other variants of vPIR outlined in the subsequent Subsection 3.2, we take the time to fix it here for the remaining part of this thesis. Note that when talking about vPIR, we expect the plain protocol of Definition 3.1 to meet all the requirements defined below it. That is, unless otherwise stated, we envision vPIR schemes to be single-server computational PIR schemes that tolerate fraudulently generated database digests, fulfill the standard correctness condition and further guarantee

integrity and malicious privacy.

To close the current subsection, we comment on the commitment and the combined verification-decryption algorithm. Apart from calling the latter simply Decryption, it is also often referred to as Verification or Reconstruction [70, 7, 13, 68]. In earlier vPIR proposals [70, 7], the verification setup was part of the query algorithm. Later works criticized that the above two schemes, among others, do not comply with integrity and claimed that, to do so, every single-server vPIR scheme requires some kind of digest from the original database [17]. This digest is either provided by a digest or hint algorithm as above [17, 62, 13] or it is part of the preprocessing stage [25, 68, 55, 58]. Assumptions on the digest generation may vary: Some schemes require honest digests while others can handle maliciously generated ones as well. For an overview, see Table 2. In [25], the term *coherence* is introduced to distance their scheme tolerating fraudulent digests from such that rely on trusted parties to compute the commitment. While we adopt their definition of coherence, we stick to the term *integrity*, since we find it more suitable.

3.2 Flavors of vPIR

Since the first scheme for verifiable PIR was introduced by Zhang and Safavi-Naini in 2014 [67], research into this topic has grown and more schemes for different settings have been proposed. In the course of this newly sparked attention for the field, different notions of PIR schemes offering varying extents of verification arose. We give a short overview of these different flavors and fix the terminology used in the subsequent parts of this thesis.
In the literature, both terms, *verifiable* and *authenticated* PIR, refer to schemes, that allow clients to validate outputs and to safely reject incorrect ones. Here, safely means that even in case the server learns about the abort the integrity of the query index is not compromised [17, 23, 58, 62, 39, 70, 13, 55]. In [23], it was suggested to call the preliminary stage of verifiable PIR that does not yet protect against selective-failure attacks, *maliciously secure*. However, this terminology has not been adopted in later works and was even redefined in [25].
The authors of [25] argued that the procedure of verification differs from authentication, where a party proves its identity, and hence objected to the usage of *authenticated* PIR. Instead, they proposed the terminology *(fully) maliciously secure*, describing schemes that defend against selective-failure attacks and allow for digests to be manipulated, too. In case the protocol relies on honest digests, they opt for the term *semi-maliciously secure* PIR.
Certain verifiable schemes fail to meet the integrity requirement and omit the initial distribution of a database digest. In this context, verifiability requires no storage of commitments to inhibit modifications of the database across queries. This is why the authors of [13] refer to such schemes as *stateless*. We believe consistency of the underlying database to be an important property of vPIR schemes and therefore do not consider this notion further.
However, the term *stateless* appears in another, more interesting context. Even though the authors of [55] acknowledged that only by means of a database commitment the honesty of the server's response can be verified, they nonetheless suggested integrating clients with no state into vPIR protocols. While these *stateless clients* may not hold material for

verification themselves, they may hide among other so-called *stateful clients*. The primary idea is to transfer security guarantees: While only stateful clients enjoy full security as promised by verifiable PIR, stateless clients may rely on them to detect malicious behavior and to incriminate deceptive servers publicly. This deterrent of publicly loosing reputation should serve as an incentive for the server to behave honestly for all queries. The term *covert* vPIR was coined for this type of scheme, where clients without states may also engage in the protocol yet only enjoy weakened security guarantees. It is evident that covert vPIRs require queries posed by stateful or stateless clients to be indistinguishable. Another distinction arises from the scope of verification: The server's honesty may either be privately or publicly verifiable. The latter notion of *publicly verifiable* schemes [7, 55] is of particular interest for covert vPIR that relies on strong incentives for the server not to cheat.

In 2022, a currently stand-alone notion of *verifiable* PIR was presented in [7]: Instead of validating the correctness of the server's response, properties of the underlying database are verified. These should either hold for the entire database or for any selection of a fixed finite number of entries. The authors speak of global or ℓ-local properties respectively, where ℓ is the number of entries the property spans. Observe that, while being more general, depending on the precise property formulation, this notion may provide much weaker guarantees than regular vPIR schemes.

To illustrate this special idea of verifiability, we give examples of properties that can be checked by schemes as specified in [7]. To this end, assume that a server holds a database of articles and claims every article to be fact-checked and digitally signed by a trusted third party. A property of interest might be whether every entry contains a valid signature. This property is 1-local, since it only concerns one item at a time. If a database passes verification with respect to this property and the source signing the entries is trusted, misbehavior on parts of the server can be detected by validating the signature of the response received. Now, to exemplify a global property, assume that the articles in the database are regularly updated. Then the database holder could relax the above guarantee and claim that, at all times, at least 90% of the entries contain valid signatures. This is a statement about the entire database and hence global.

For a broader perspective, we also state two further notions of vPIR schemes that only occur for multi-server protocols. While in the single-server setting, to the best of our knowledge, it is only possible to detect manipulations, the multi-server case might also allow for reconstructions of the correct answers. For this, at least one of the servers needs to be honest and serve as the ground truth to compare the responses against. When reconstruction is possible, the scheme is said to provide *full security* [17]. Unfortunately, this again conflicts with the previous notion of fully maliciously secure schemes [25]. Moreover, some works investigate *robust* PIRs (e.g. [4, 22]). Their main incentive is to guarantee answers even in the presence of deviations from the protocol or system failures. We again stress that this work, however, focuses on the single-server setting.

3.3 Related Work on vPIR

For an overview of existing vPIR schemes and their variants, we summarize recent works in Table 2 without claiming completeness. As criteria we use the key idea of how verifiability is realized, state any mathematical assumptions and other functionalities the schemes

are based on and describe the scope of protection they provide in an abstract manner. Further, we list their asymptotic communication costs and also state the computational efforts wherever they are known. Other noteworthy aspects of the schemes are collected in the Miscellaneous column.
We primarily sort the entries according to their scope of protection, ranging from weak to strong. The weakest schemes are susceptible to selective failure attacks. Stronger ones are divided into those that do not tolerate maliciously generated digests and those that do. For the latter, one can further regard whether or not the server is forced to keep to a consistent database across several queries. If the scope of protection is not specified in such a nuanced way in the papers, we associate schemes with the strongest level they are known to guarantee. This may, however, be weaker than the actual protection they provide. To indicate the different groups, we use two types of lines: Dotted lines separate schemes achieving the same level of protection and solid lines introduce groups with stronger security guarantees. In case the protection guarantees are the same, we consider the communication costs of the schemes in the online phase, that is, the costs when the actual query is sent. Here, we arrange the protocols in increasing order, starting with cheaper and ending with more expensive ones.
For Table 2, let n be the database size and m be the bit size of its entries. If arranged as a matrix, let r and c be the number of rows and columns of the reshaped database, respectively. Note that oftentimes, $n = r$ and $m = c$. As usual, λ is the security parameter. Since several abbreviations appearing in the table relate to concepts that are not introduced in this thesis, we remind the reader of Table 1 that lists references on them.
Most of the schemes presented in Table 2 ensure verifiability by either leveraging some kind of commitment protocol or by introducing test queries. Commitments (or digests) bind the server to a specific database while additional queries conceal the actual request and test the server's honesty by checking its responses against expected values. In case these so-called *dummy queries* are not chosen deliberately but rather generated randomly, they may, however, fail to fulfill their purpose. This is because the client then lacks sufficient data to verify answers to these queries meaningfully. Hence, even wrongful responses are often accepted whereas rejections only occur for the actual queries. Thus, selective-failure attacks may still apply [62].
While most of the works presented in Table 2 take the approach of constructing concrete vPIR schemes, it is noteworthy that there are also generic compilers that can transform any PIR into a vPIR scheme. Among them are the proposals of [39] and [25].

Table 2: Overview of existing vPIR schemes.

Scheme	Main Ideas	Assumptions	Scope of Protection	Costs	Miscellaneous
vPIR from LWE [70]	• hide query between duplicates of the actual query, zero queries and random queries arranged as a query matrix • order of query types only known to the client • expected outcomes – same answer for redundant queries – checksum of zero queries = 0 – checksum of arbitrary queries $\neq 0$	binary LWE	• no database commitment • vulnerable to selective-failure attacks	• communication: $\mathcal{O}(C \cdot \mathsf{poly}(\lambda)\log(\lambda))$ (constant C depends on the number of supported additions) • computation: $\mathcal{O}(\mathsf{poly}(\lambda))$	• additive HE scheme • number of obfuscating queries depends on security parameter • supports single and multi-bit databases
Verifiable PIR [7]	• construction based on batch arguments for the index language	LWE, existence of poly-log PIR schemes of complexity C	• no database commitment • vulnerable to selective-failure attacks • verification of predicates, local or global, sublinear in the database size	• ℓ-local vPIR: $\mathcal{O}(c \cdot \mathsf{poly}(\lambda, \log(r)))$ (communication) • global vPIR: $\mathcal{O}(C \cdot c \cdot \mathsf{poly}(\lambda, \log(r)))$ (communication)	• general vPIR construction • verification of properties of the database • for global properties, database has to be independent of the query location
Crust [62]	• database digest for consistency • precompute plain and randomized hints (= sums of subsets of database entries) • replace indices of interest by *crumbs*, i.e., random index together with offset • outcome = hint - (response + crumbs)	pseudorandom (one-way) function	• selective-failure secure • disallows malicious digests	• communication: $\mathcal{O}(n)$ (offline), $\mathcal{O}(\sqrt{n} \cdot (m + \lambda))$ (online [25]) • computation: $\mathcal{O}(n \cdot \lambda)$ (offline), $\mathcal{O}(\sqrt{n} \cdot (m + \lambda))$ (online [25])	• per offline phase, $\sqrt{n}$ queries can be posed • blockwise streaming of the entire database for hint generation

continued on the next page

Table 2 – *continued from the previous page*

Scheme	Main Ideas	Assumptions	Scope of Protection	Costs	Miscellaneous
EAPIR [58]	• same as for aPIR (DDH) • hash indices and reshape database according to hash buckets • padding to mask bucket lengths	DDH, random oracle (hashes)	• selective-failure secure • disallows malicious digests	• asymptotically: same as for aPIR • concretely – communication: 1,1 times less than aPIR – computation: 1,2 times faster	• three parties: client (honest), database owner (honest), cloud server (malicious) • index search restricted to hash bucket, hence very efficient • database owner sees actual query • client not involved in the preprocessing phase
aPIR (LWE) [17]	• digest for each database column • answer is verified against all column digests	LWE	• selective-failure secure • disallows malicious digests	• communication: $\mathcal{O}(\sqrt{n} \cdot m \cdot \lambda)$ [25] • computation: $\mathcal{O}(n \cdot m \cdot \lambda)$ [25]	• single-bit databases • database digest may be public
aPIR (DDH) [17]	• digest for entire database • infeasibility of producing the same digest with a different database	DDH	• selective-failure secure • disallows malicious digests	• communication: $\mathcal{O}(\sqrt{n} \cdot m \cdot \lambda)$ [25] • computation: $\mathcal{O}(n \cdot m \cdot \lambda)$ [25]	• single-bit databases; adaptable to multi-bit entries • database digest may be public
vPIR resisting malicious cloud servers [68]	• parity checksum for each database entry • public global checksum as ground truth	infeasibility of computing inverses of large matrices	• selective-failure secure • disallows malicious digests	• communication: $\mathcal{O}(n \cdot m)$ (offline), $\mathcal{O}(n \cdot (\log_2(n) + m))$ (online)	• three parties: client (honest), database owner (honest), server (malicious) • one-time initialization

continued on the next page

Table 2 – *continued from the previous page*

Scheme	Main Ideas	Assumptions	Scope of Protection	Costs	Miscellaneous
vHE-PIR [71]	• based on basic scalar product between queried incidence vector and database • zk-SNARK proof using Quadratic Arithmetic Program (QAP) representing the basic functionality • optimized for packed ciphertexts; batched query and database structure	zk-SNARKs, arithmetic circuits, QAP, decisional LWE	• selective-failure secure • ignorant to malicious digests (private server input)	• communication: $\mathcal{O}(n)$ • computation: $\mathcal{O}(\log(n))$	• private setup by client • no data preprocessing, no commitment • suitable for matrices • integrity relies on statistical argument (Schwartz-Zippel)
fully malicious aPIR [23]	• initial digest validation phase • indistinguishability of validation and actual queries • answer extractor constructable from validation phase • extractor produces accepting answers for all queries	DDH	• selective-failure secure • allows malicious digests • inconsistency of database across queries possible	• communication: $\mathcal{O}(\sqrt{n} \cdot m \cdot \lambda)$ (both, validation and online phase [25]) • computation: $\mathcal{O}(n \cdot m \cdot \lambda)$ [25]	• one-time and single round-trip validation phase • validation requires no server modification • low client-side storage costs
Multi-query vPIR [39]	• PIR answers extended by SNARK proofs	(a-)PIR (complexity C), SNARKs (proof size p)	• selective-failure secure • allows maliciously computed SNARK proofs	• communication: $\mathcal{O}(C \cdot \#\text{queries} + p)$	• generic constructions: PIR $\rightarrow$ vPIR, aPIR $\rightarrow$ multi-query vPIR • multi-query property requires consistent databases • no preprocessing phase

continued on the next page

Table 2 – *continued from the previous page*

Scheme	Main Ideas	Assumptions	Scope of Protection	Costs	Miscellaneous
vPIR almost for free [25]	• encoded database can tolerate errors (uses locally-decodable codes (LDC)) • vector commitment for database consistency	PIR, LDCs, vector commitments	• selective-failure secure • allows malicious digests	• communication: $\mathcal{O}(n^{\varepsilon} \cdot \lambda(m + \lambda \log(n)))$ (online) • computation: $\mathcal{O}(n \cdot \log(n))$ (offline), $\mathcal{O}(n \cdot (m + \lambda \log(n))$ (online)	• doubly-efficient PIR variant: costs are poly$(\log(n), \lambda)$ each • generic construction PIR $\rightarrow$ vPIR • no communication in the preprocessing phase
VeriSimplePIR [13]	• server bound to database by means of a (non-interactive) commitment scheme • verification proof reusable until the first response rejection • challenge for commitment scheme hidden from the server	LWE, SIS, random oracles (hashes)	• selective-failure secure • allows malicious digests • consistent database guaranteed	• communication: $\mathcal{O}(\sqrt{n} \cdot \lambda)$ (offline), $\mathcal{O}(\sqrt{n} \cdot (m + \lambda))$ (online [25]) • computation: $\mathcal{O}(n)$ (offline), $\mathcal{O}(n \cdot (m + \lambda))$ (online [25])	• vLHE for de-/encryption • aborts trigger recomputation of preprocessing phase • (almost) no overhead for honest servers • strict validation: only exactly correct responses pass • large local storage required
vPIR with small client storage [55]	• commitment scheme generates a database digest together with a proof • digest for consistency and decryption • proof for verification	(R)LWE, SIS	• selective-failure secure • allows malicious digests • consistent database guaranteed	• communication: asymptotically equal to VeriSimplePIR [13]	• covert and symmetric variants • efficient without needing Fiat-Shamir heuristic or random oracles • LHE based • black-box verification of HE operations: verification after rather than before decryption

4 Verifiable FHE

Having set up all basic definitions for vPIR and sketched its existing nuances, we now proceed similarly for the concept of vFHE. First, we work towards a general definition of verifiable FHE which is again based on a set of algorithms that govern the interaction between the delegating and the computing party. Like before, vFHE schemes usually fulfill certain requirements, which we also define. Then, a broader discussion on different levels of security emerging in the context of vFHE follows. After introducing some (v)FHE specific terminology, we mention technical challenges that have to be overcome when constructing schemes concretely. This leads to presenting a classification of vFHE schemes based on the mechanisms used for verification. Finally, we again provide an overview of concrete vFHE protocols in Table 4.

4.1 Defining General vFHEs – Algorithms and Requirements

We briefly motivate our definition of general vFHE schemes. Just like vPIR extends regular PIR schemes by adding a verification mechanism, vFHE builds on the cryptographic primitive of FHE. The decisive property of FHE is its homomorphic nature: It enables operating on encrypted data without having to reveal the underlying plaintext. In particular, secret data can first be encrypted, then be outsourced to any computing party and only later, after evaluation, be decrypted again in a trusted domain. However, when assuming malicious instead of semi-honest servers, it is not only of concern how to preserve the confidentiality of the underlying data [43] but also how to ensure the correctness of the computed result. Therefore, it is of interest to impose stricter requirements on FHE schemes. In verifiable FHE, a new layer of protection is added. While still maintaining confidentiality, there are now also techniques incorporated to check the correctness of the outcome. We will formalize this incentive to allow for integrity as well by setting up a basic framework for verifiable FHE schemes, settling on the defining algorithms and requirements. For this, we mainly follow the notation and structure of [60] for two main reasons: Their generality in terms of the definition of a vFHE scheme (others view vFHE as a by-product of integrating so-called verifiable computation into regular FHEs [29, 10, 63]) and their modularity. They deliberately separate confidentiality and integrity related requirements to allow for a clear transition from FHE to vFHE and to define extensions to their basic vFHE protocol more easily. We remark that in the following we only consider public-key encryption. This is because most work on FHE assumes this setting and it is more convenient to stick to one type of encryption rather than switching back and forth between the public- and the symmetric-key variants. Consequently, encryption oracles are trivially available to adversaries and will not be stated explicitly.

Definition 4.1 (Verifiable Fully Homomorphic Encryption - vFHE [60])**.** A *verifiable Fully Homomorphic Encryption scheme (vFHE)* is a tuple of PPT algorithms $(\mathsf{KGen}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{Vf}, \mathsf{Dec})$ defined by

- $\mathsf{KGen}(1^\lambda, f) \to (\mathsf{pk}, \mathsf{sk})$: Based on the security parameter λ and a function f, the *key generation* algorithm generates a public key pk and a secret key sk, for en- and decryption, respectively. While pk is published, sk is only given to the client.

- $\mathsf{Enc}_{\mathsf{pk}}(x) \to (c_x, \tau_x)$: The *encryption* algorithm takes a private function input x and uses the public key pk to produce a ciphertext c_x together with some auxiliary information τ_x.

- $\mathsf{Eval}_{\mathsf{pk}}(c_x) \to (c_y, \tau_y)$: In the *evaluation*, the function f is homomorphically applied to the ciphertext c_x to obtain an encryption of $y := f(x)$, namely c_y, and some auxiliary information τ_y.

- $\mathsf{Vf}_{\mathsf{sk}}(c_y, \tau_x, \tau_y) \to b$: Using the secret key sk and the auxiliary information τ_x and τ_y, the *verification* procedure sets an acceptance bit b. If c_y is found to be truthful, set $b = 1$, otherwise, set $b = 0$ and reject the response.

- $\mathsf{Dec}_{\mathsf{sk}}(c_y) \to y$: In case $b = 1$ in the verification step, the *decryption* recovers the function evaluation $y = f(x)$ from the ciphertext c_y using the secret key sk.

Remark 4.2. There are several aspects we want to draw the reader's attention to.

- **Evaluation input** Typical FHE applications require the evaluation of multivariate functions, which usually take multiple ciphertexts as input. However, in the above protocol, the evaluation algorithm Eval only has a single input, namely c_x. We opted for this notation, since we do not know the exact number of inputs to the function f in advance. Hence, we simply consider c_x to be the set of all ciphertexts that the client provides to the homomorphic function evaluation. In particular, c_x is not restricted to a single ciphertext.

- **Function dependency of key generation** The literature disagrees on whether the function f is an input of the key generation function or of the evaluation algorithm. Historically, the first verifiable FHE schemes arose from combining standard FHEs with Verifiable Computation (VC) to overcome the lack of integrity assurances when using plain FHEs [29, 69]. While VC usually produces function dependent key pairs, FHE keys are function independent [10, 29]. In [49], this distinction is nicely highlighted by generating two separate key pairs in KGen, one for the VC and one for the FHE component. While we adopt this distinction when giving a more detailed and practical definition of vFHE for constructing vPIRs in Section 6, for the general vFHE blueprint we only introduce a single key pair. This has the advantage of keeping the general definition short while already giving an intuition for which algorithms are based on public and which on secret information. Yet, we emphasize that the generation of the FHE keys alone is function independent.

- **Server inputs** In real-world applications, the function f may also take inputs from the server. This is already supported in the above definition, since public key encryption is assumed. Hence, just like clients, the server can encrypt its data and provide the resulting ciphertexts as additional inputs to the evaluation algorithm Eval. If, however, the server should be able to provide private inputs, the scheme requires adaptation. We refer to Definition 4.6 (vFHE with Server Privacy) in [60] for details.

- **Common terminology**

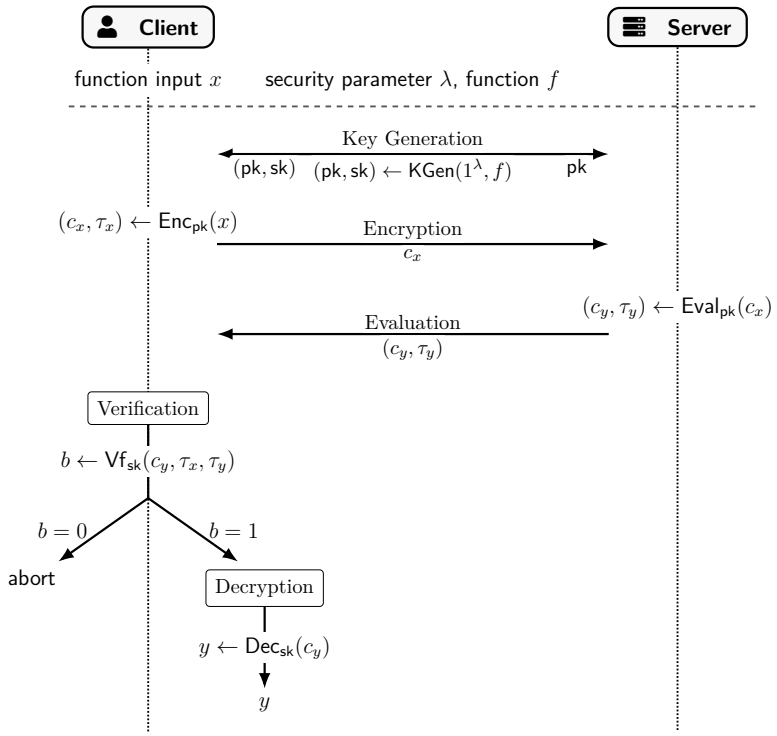

Figure 6: Interactive protocol for the general vFHE blueprint.

- Some schemes only support client-side en- and decryption and therefore utilize a symmetric-key encryption scheme. However, it is more common to consider encryption to be public-key, which is also the original context FHE was introduced in [31]. The corresponding schemes are called *publicly delegable* [60].
- Depending on whether a ciphertext was produced by Enc or Eval, one distinguishes two types: *Fresh ciphertexts*, like c_x, are the outputs of the encryption algorithm; *evaluated ciphertexts*, like c_y, are the outputs of Eval.
- Definition 4.1 only offers *designated verifiability*. This means that secret material is required to verify ciphertexts. However, the authors in [60] claim that it can easily be extended to public verifiability by simply using a publicly verifiable analogue of the verification component.

Like for vPIR, we visualize the vFHE protocol by means of the diagram in Figure 6. The algorithms in Definition 4.1 have to fulfill the subsequent requirements.

Definition 4.3 (Correctness for vFHE [60, 15])**.** A vFHE scheme is *correct*, if honest computations decrypt to the expected result with high probability. Formally, let f be any function, $x \in \mathrm{Dom}(f)$ be any input. Then, for a vFHE scheme to be correct, the following probability should be negligible

$$\Pr\left[\mathsf{Dec}_{\mathsf{sk}}(c_y) \neq f(x) \;\middle|\; \begin{array}{c}(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KGen}(1^\lambda, f) \\ (c_x, \tau_x) \leftarrow \mathsf{Enc}_{\mathsf{pk}}(x) \\ (c_y, \tau_y) \leftarrow \mathsf{Eval}_{\mathsf{pk}}(c_x)\end{array}\right] \leq \mathsf{negl}(\lambda).$$

If the probability equals zero, we speak of *perfect correctness*, following the terminology in [15].

Remark 4.4. Refinements of the correctness definition exist which explicitly distinguish between correct decryption of fresh and evaluated ciphertexts, or that allow for evaluation results to differ slightly from the actual outcome. While the distinction can be mirrored by setting $\mathsf{Eval} := Id$ when fresh ciphertexts are considered, where Id denotes the identity function, allowing for minor deviations in the decryption outcome is a true extension to the given definition. We refer the interested reader to [15] for a more extensive breakdown and precise definitions.

To rule out trivial (v)FHE schemes, where the evaluation simply forwards all computational effort to the decryption function, the bit size of ciphertexts should be bounded. In particular, the size should only depend on the security parameter and not on the evaluated function f or the number of arguments it takes. As a consequence, performing homomorphic operations also maintains a moderate growth of the ciphertext size [2]. For standard FHE schemes, this idea is formalized in the compactness requirement [31]. When adding verification, care has to be taken: To verify that some specified function f has correctly been evaluated, it is usually required to have some description of f at hand. This means, however, that verification depends on the circuit size. With this in mind, we base our new definition of compactness for vFHE on the proposal of [1], that identified this discrepancy.

Definition 4.5 (Compactness for vFHE [2, 10, 1])**.** A vFHE scheme is *compact* if the bit size of fresh and evaluated ciphertexts only depends on the security parameter. Concretely, let λ be any security parameter. Then, compactness requires that for every function f, every input x and key pair $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KGen}(1^\lambda, f)$, the size of all ciphertexts $(c_x, \_) \leftarrow \mathsf{Enc}_{\mathsf{pk}}(x)$ and $(c_y, \_) \leftarrow \mathsf{Eval}_{\mathsf{pk}}(c_x)$ is polynomially bounded in λ, i.e.,

$$|c_x|, |c_y| \leq \mathsf{poly}(\lambda).$$

Importantly, no bounds are imposed on the verification algorithm.

Definition 4.6 (Completeness for vFHE)**.** A vFHE scheme is *complete*, if Vf always accepts honestly computed results. Formally, let f be any function and $x \in \mathrm{Dom}(f)$ be any input. Complete vFHE schemes satisfy the following condition

$$\Pr\left[\mathsf{Vf}_{\mathsf{sk}}(c_y, \tau_x, \tau_y) = 1 \;\middle|\; \begin{array}{r}(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KGen}(1^\lambda, f)\\ (c_x, \tau_x) \leftarrow \mathsf{Enc}_{\mathsf{pk}}(x)\\ (c_y, \tau_y) \leftarrow \mathsf{Eval}_{\mathsf{pk}}(c_x)\end{array}\right] = 1.$$

Definition 4.7 (Soundness for vFHE)**.** A vFHE scheme is *sound*, if Vf only accepts honestly computed results; incorrect responses are rejected with high probability. Formally, let f be any function, $x \in \mathrm{Dom}(f)$ some input and $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ any PPT adversary. Then soundness asks for

$$\Pr\left[\begin{array}{l}\mathsf{Vf}_{\mathsf{sk}}(c_y, \tau_x, \tau_y) = 1\\ \wedge\, \mathsf{Dec}_{\mathsf{sk}}(c_y) \neq f(x)\end{array} \;\middle|\; \begin{array}{r}(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KGen}(1^\lambda, f)\\ x \leftarrow \mathcal{A}_1^{\mathcal{O}_{\mathsf{Vf}}}(\mathsf{pk})\\ (c_x, \tau_x) \leftarrow \mathsf{Enc}_{\mathsf{pk}}(x)\\ (c_y, \tau_y) \leftarrow \mathcal{A}_2^{\mathcal{O}_{\mathsf{Vf}}}(c_x, \tau_x)\end{array}\right] \leq \mathsf{negl}(\lambda).$$

For security, we adapt the definition provided in [60] in two ways: First, to the best of our knowledge, current FHE schemes do not satisfactorily achieve notions beyond IND-CPA

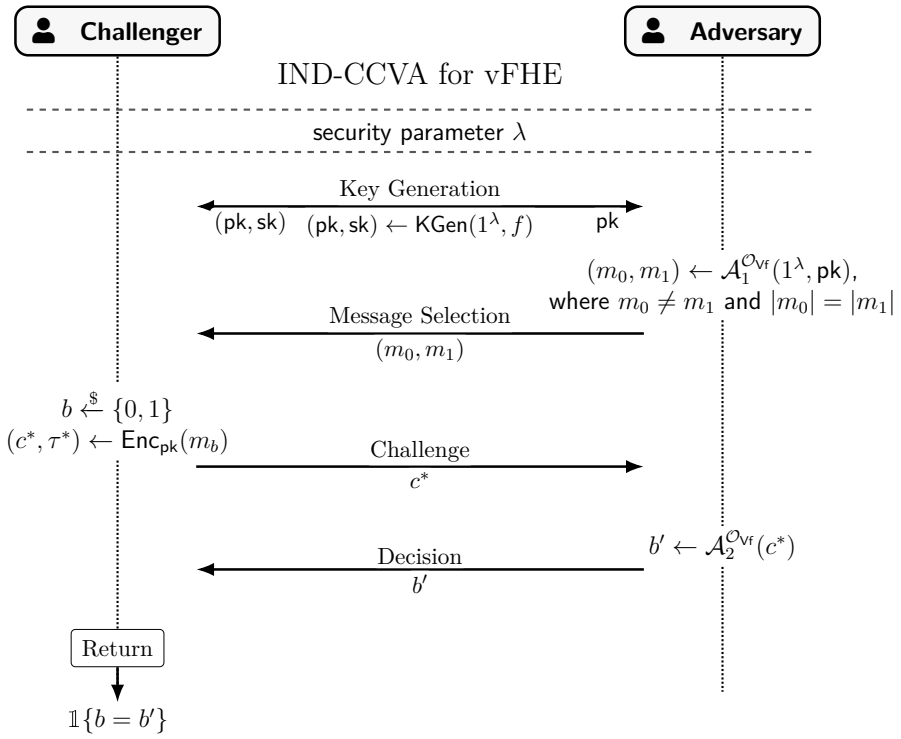

Figure 7: IND-CCVA security game for vFHE.

security [27]. Therefore, we refrain from equipping attackers with decryption oracles. Second, adversaries should gain no information from the verifier's decision to accept or reject a ciphertext. This is mirrored by providing access to verification oracles throughout the game and is captured by the notion of *INDistinguishability under Chosen-Ciphertext Verification Attacks (IND-CCVA)*, proposed in [44]. For better readability, we present the concrete definition of the IND-CCVA game in Figure 7. Note that apart from minor notational adaptations for the context of vFHE, it is the same indistinguishability game as the one provided for security of general encryption schemes in Figure 1. Now, the oracles in the blueprint are instantiated with $\mathcal{O}_1 = \mathcal{O}_2 = \mathcal{O}_{\mathsf{Vf}}$.

Definition 4.8 (Security for vFHE [50])**.** Let IND-CCVA be the game as defined in Figure 7. Then, a vFHE scheme with security parameter λ is IND-CCVA secure, if for any PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, the attacker's advantage of winning the game is negligible. That is, we ask for

$$\mathsf{Adv}_{\mathcal{A}}^{\text{IND-CCVA}}(\lambda) \leq \mathsf{negl}(\lambda),$$

where the advantage $\mathsf{Adv}_{\mathcal{A}}^{\text{IND-CCVA}}(\lambda)$ is defined analogously to Definition 2.4.

There are several ways to define security for verifiable FHE and we acknowledge that the definition above may not be the most standard one. Yet, we justify our choice by noting that first, the presence of verification oracles will become essential for the vPIR construction we propose and second, we want the definition to be as permissive as possible while still providing meaningful security guarantees. To contextualize our definition, we provide an overview of other security notions for vFHEs.

Security Levels of (v)FHEs In the introduction in Section 1, we have already depicted the necessity of security guarantees when using vFHEs in practice. Further, the main notions of security for encryption schemes are presented in the form of indistinguishability games in Subsection 2.1. Now, we focus more on the concrete setting of vFHE by elaborating on the scope of security in this context and by detailing tailored attacks such protocols have to safeguard against.

Generally speaking, attacks can be directed against one of two properties: Correctness or confidentiality. We start with turning our attention to the first. While general FHE schemes are designed to protect the client's privacy, assurances of correctness come with the incorporation of verification mechanisms. To target these guarantees, the server can try to deform inputs in such a way that they result in valid ciphertexts but lead to wrong or meaningless answers after decryption. This can, for example, be caused by intentional noise overflow. One speaks of *application-level correctness* when expressing that vFHE responses are not only valid but also compliant with the intended functionality [60].

Before addressing the second type of attack, namely compromising the confidentiality of data, we remark that such attacks are not special to verifiable FHE but rather FHE or even encryption schemes in general. The reason we still want to give more insights is that these breaches are a rather serious issue and motivate the construction of robust FHE protocols. Critical in this context are *key-recovery attacks* that exploit the malleability of ciphertexts to gradually learn the secret decryption key. Hereby, the malleability is a direct consequence of the homomorphism property and allows computing parties to make deliberate alterations to the ciphertexts. These parties can then observe the client's reaction upon decryption, a procedure that heavily relies on the secret key. Thus, failing to decrypt can leak information about the key. By challenging the client with such tampered ciphertexts repeatedly, an adversary can, in the worst case, eventually learn not just parts but the entire secret key. Note that operations using evaluation keys are also prone to this kind of attack since these keys, too, encapsulate secret data [60]. In this context, the authors of [60] identified a shortcoming of current assumptions in the literature: Oftentimes, only verification oracles and not the stronger notion of decryption oracles are considered for the integrity guarantees. Protections against verification oracles are often weaker, since the data used to verify answers usually does not disclose information about the secret data directly. Yet of course, this is not true in all cases and sometimes the data revealed suffices to successfully decrypt ciphertexts [44]. In the scenario described earlier, however, the client's reaction is exploited as a full power decryption oracle and one would therefore need appropriate defenses against this type. We point out that, unless the client responds to all ciphertexts in the same way, its behavior can always be turned into some kind of oracle. Since it is implausible that there is no correlation between the client's reaction and the correctness of the answer, having oracles is unavoidable [43].

This finding motivates the following discussion on different security notions that have been proposed for vFHE schemes. Crucially, protocols should be secure even if verification oracles are accessible. This is addressed in [44] and [21] and we account for it by requiring our vFHE schemes to be IND-CCVA-secure (see Definition 4.8). Before presenting other approaches of integrating verification oracles into the indistinguishability game blueprint (see Figure 1) and outlining the resulting enhanced security guarantees, we first elaborate on the standard classification of security into IND-CPA, IND-CCA1 and IND-CCA2 in the context of vFHE.

As mentioned before, as of now, FHE schemes seem to be restricted to establishing IND-CPA security [27]. Consequently, this security level is a natural starting point for defining the security of verifiable FHE schemes. Yet recent years have seen advances in the scope of security provided. At first, it seems tempting to ultimately achieve IND-CCA2 security but this was proven to be impossible [64]. In fact, the notion is in direct contradiction to the malleability property of homomorphic operations. This is due to the following permissible strategy: An adversary can choose some invertible function and apply it to the challenge ciphertext c^* obtained during the Challenge step of the indistinguishability game. It then asks the decryption oracle $\mathcal{O}_2$ to decrypt this altered ciphertext. By evaluating the inverse of the chosen function on the result, the adversary gets the plaintext corresponding to c^*. This breaks IND-CCA2 security [64].

Therefore, works aiming at stronger security than IND-CPA either attempt to establish IND-CCA1 security [60, 43] or come up with intermediate security notions that lie between IND-CCA1 and IND-CCA2 [15, 50, 64]. Indeed, to strengthen IND-CCA1 security further while remaining strictly weaker than IND-CCA2, it is the next logical step to integrate verification oracles: Unlike decryption oracles, they may be queried with every ciphertext, even with the challenge c^* [44, 21]. Hence, they can remain accessible in the post-challenge phase of the IND-CCA1 security game. This is desirable as it combines the guarantees of IND-CCA1 with protections against verification attacks. The term *IND-CCA1.5* [21] was coined for this level of security and in [50] it is related to other existing scopes of security. Some of these are, like IND-CCA1.5, stronger than plain IND-CCA1. We look more closely into the proposals of [50, 15] and [64]. Their work comes with the benefit of not only introducing new security notions in theory, but also presenting schemes that implement them.

In the first two papers, security against *verified Chosen-Ciphertext Attacks (vCCA)* is ensured, a notion introduced by [50]. The main idea of vCCA is to analyze ciphertexts and to detect, whether they were computed from the challenge ciphertext c^* or not. Only if this question is answered negatively, the post-challenge oracle $\mathcal{O}_2$ decrypts. The check is facilitated by an extraction algorithm $\mathsf{Extract}$: If an input ciphertext is the result of some evaluation performed by Eval, the algorithm recovers the ciphertexts it was computed from. An adapted decryption oracle then utilizes the $\mathsf{Extract}$ algorithm to decide whether or not to decrypt a given ciphertext. More specifically, for a queried ciphertext c, $\mathsf{Extract}$ first checks if it was computed from the challenge ciphertext c^*. If yes, the oracle aborts. Otherwise, it decrypts c as usual. With this oracle at hand, the game for *INDistinguishability under verified Chosen-Ciphertext Attacks (IND-vCCA)* is then simply defined by setting $\mathcal{O}_1 = \mathcal{O}_{\mathsf{Dec}}$ and $\mathcal{O}_2 = \mathcal{O}_{\mathrm{vCCA}}$ in the security game blueprint of Figure 1. For the concrete formalism, we refer the interested reader to [50].

The third work mentioned above, that is [64], takes a slightly different approach. While it keeps to the main idea of restricting the input to the post-challenge oracle by prohibiting the query of challenge-correlated ciphertexts, adversaries should testify to this themselves: When querying the oracle, they not only have to provide the desired ciphertext, but also submit all ciphertexts used to generate it. Then, a verification algorithm publicly verifies whether the additional ciphertexts truly evaluate to the actual query. Importantly, adversaries are not assumed to be honest when submitting inputs. The authors call their notion of security *Input-Verifiable CCA (IV-CCA)* and prove that it is implied by vCCA while being strictly weaker. The scheme they construct to achieve IV-CCA security has

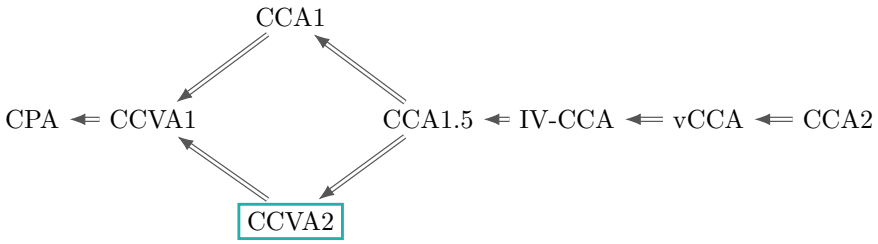

Figure 8: Comparing security notions for vFHE.

the advantage of relying on the LWE assumption alone. This shows that IV-CCA secure schemes exist in the standard model, whereas no analogous conclusion can be drawn for IND-vCCA secure protocols since the current schemes are based on stronger assumptions like SNARKs [50] or LOH [15].

For a better overview of all security notions mentioned, we provide Figure 8, showing the connection between different levels. It combines similar diagrams presented in [50, 21] and [64] but is restricted to the notions we refer to in this thesis. Highlighted is the security level that we used in Definition 4.8 to formulate the security requirement on general vFHE schemes. Note that in the diagram, IND-CCVA is split into the two separate notions of IND-CCVA1 and IND-CCVA2. This should resemble the distinction between IND-CCA1 and IND-CCA2: In both cases, the first, non-adaptive variant only allows for decryption (respectively verification) oracles in the pre-challenge phase, while the second variant provides access to the oracle throughout the entire indistinguishability game. What we previously termed IND-CCVA security therefore corresponds to IND-CCVA2. Recall that IND-CCA2 is itself not achievable by vFHE schemes but is included in the overview for completeness. Further, we point out that, apart from the implication "vCCA $\Rightarrow$ IV-CCA", all other implications have formally been proven to be strict, i.e., the other directions do not hold. To save space, we omit writing "IND" in front of all levels of security.

We close this section by recommending the reference [56] for an in-depth discussion on security guarantees in the context of FHE. Not only does it provide an encompassing overview of existing security notions for FHE and proposes even more nuanced variants, but it also outlines construction blueprints for schemes achieving these respective notions. It captures many of the important findings on the subject of security for FHE known to date. Since this topic, however, is not at the heart of our work, the brief excursion into the vast landscape of security notions presented above suffices for our own purposes.

Features of (v)FHEs While the requirements of Definitions 4.3 to 4.8 are essential for defining meaningful vFHE protocols, there are multiple additional desirable features that vFHEs may have. Below, we outline some of the more popular ones. Note that the list is not limited to extensions arising in the context of verifiable FHE, but FHE in general.

The term *fully* in fully homomorphic encryption refers to the ability of the (v)FHE scheme to handle any function f that can be expressed as a Boolean or arithmetic circuit. While it is desirable to support all functions, some schemes only work for certain classes thereof. For example, *linear* HEs only allow for linear operations like addition and scalar multiplication, while *somewhat* HE schemes only allow for a limited number of homomorphic operations until the noise grows too large [31]. When specifying the depth of the circuits

that can be evaluated in advance, one speaks of *leveled* HE schemes.
Verifiability may either be public or private. In case private data is required, a scheme is said to be *designated verifiable* [10].
Similarly, encryption might rely on private or purely public parameters. Hence, for encryption schemes, we distinguish between *private* or *public delegation* [10].
Sometimes, the encryption of inputs depends on the function that should be evaluated on them. In contrast, *multi-function* schemes have the benefit that several functions can be performed on the same inputs while these only have to be encrypted once [10].
Outsourceability describes the intention of relieving clients of heavy computations by delegating them to external parties, like servers. On the one hand, providing the function inputs and verifying the obtained answers should be substantially cheaper than carrying out the function computation. On the other hand, the computing party should roughly have the same costs as evaluating the function as-is, without any verification mechanisms embedded. In short, ideal client and server computation times are $\mathcal{O}$(input size + output size) and $\mathcal{O}(|C_f|)$, respectively, where C_f is a circuit evaluating f [29].
The setting of outsourced computation typically assumes that the client provides all inputs x as well as the desired function f to an untrusted server that is then merely tasked with homomorphically evaluating $f(x)$. In practice, however, the server might also choose inputs or define the functionality. Therefore, schemes can be categorized by their support of server inputs and whether they are private or public. Similarly, f might either be public knowledge, private to the client or private to the server [60]. Needless to say, each setting comes with its own advantages but also challenges.
Some contexts rely on the incorporation of randomness, the use of non-arithmetic operations, like logical or bitwise operations, or include secret parameters. To perform such tasks homomorphically, non-deterministic computations must be supported as well. These can be modeled by allowing additional inputs from a third party [10].
For a quick lookup, we organize the above features of vFHEs in Table 3.

Table 3: Overview of (v)FHE features.

Feature	Meaning
fully HE	supports arbitrary functions
linear HE	supports linear operations (addition, scalar multiplication)
somewhat HE	supports a limited number of operations
leveled HE	supports a prespecified number of operations
designated/public verifiability	private/only public data used for verification
multi-function	encryption is function independent
private/public server inputs	supports respective type of server inputs
client private/server private/public function	supports respective type of function f

Later in this thesis, we are particularly interested in verifiable FHE in the context of verifiable PIR. Hence, for our setting we focus on schemes where the client provides inputs

in the form of queries and the server contributes a database that is usually assumed to be public. For design reasons, we model the database as internal knowledge of the server that concretizes the functionality. This allows us to view the protocol as one that takes inputs from a single party, which is the client. Note that it suffices to support a single functionality, namely the PIR protocol itself. This is typically public knowledge, so we do not need to consider private circuits. Most commonly, vPIRs are designated verifiable, so we concentrate on this notion as well.

4.2 Construction Challenges and Taxonomy of vFHE

Existing literature has identified several challenges when constructing verifiable FHE schemes, especially when taking real-world environments and applicability into consideration. Below, we mention a selection of some common obstacles.
Recent works attest that FHE schemes have improved significantly in performance since their introduction; today, they are even used in practice [43]. Yet, some issues like expensive computation persist or arise from methods that are specifically tailored to ensure the efficiency of the underlying standard FHE schemes [14].
Among the first kind are large ciphertext spaces and respective representations [10, 60]. Note that there are, in fact, already ideas to mitigate this problem: Since the ciphertext size is heavily influenced by the choice of the underlying mathematical problem [60], basing the scheme on a different similarly hard assumption might already suffice as a solution. Furthermore, one can resort to tools such as homomorphic hashes [10] to reduce the ciphertext space. However, these solutions often have caveats on their own like largely unsupported operations or additional assumptions.
Depending on the techniques applied for turning general FHEs into vFHEs, a second problem arises from the ciphertext representation: Oftentimes, message spaces and their inherent arithmetic are not compatible [10, 60, 14], resulting in costly type conversions.
Moreover, many techniques like standard MACs or signatures are not suitable for ensuring integrity in the context of vFHEs. This is because their functioning tolerates no degree of malleability, which is at the core of homomorphic encryption [43].
What also hinders the construction of new verifiable FHE schemes, is that many ideas used in other works cannot be generalized easily. Instead, optimizations and integrity integrations are often tailored to one specific underlying FHE scheme [43]. Even if the main idea allows generalization, it might not always be beneficial: Many vFHEs, for example, rely on some form of argument of knowledge to establish integrity. This is a rather strong assumption, making it inapplicable in cases asking for milder assumptions [15].

With these challenges in mind, we now present a taxonomy of vFHE schemes introduced in [60, 43]. It classifies existing protocols according to their main technique for establishing verifiability. Three types of approaches were identified: Homomorphic message authentication codes, zero-knowledge proofs and trusted execution environments. We briefly define all three of them and outline their application within vFHEs, following the structure in [60].

Homomorphic MACs (HMACs) are unforgeable tags to messages which attest to their integrity and authenticity. To verify results computed from tagged cipher- or plaintexts, a

client typically precomputes the function over tag-related pseudorandom challenge values. In the online phase, it then merely has to compare the precomputed data to parts of or simple calculations from the actual response [14]. There are three stages at which HMACs can be included in the encryption process: They can either be computed in parallel, before or after the encryption, leading to the methods of *Encrypt-and-MAC (EaM)*, *Encrypt-then-MAC (EtM)* and *MAC-then-Encrypt (MtE)*. For EaM, both the encryption algorithm and the HMAC operate on the same underlying plaintext. The tag and ciphertext are then processed in parallel but independently. On the one hand, this requires the homomorphic MAC to come with strong security guarantees itself since it is directly based on the plaintext [60, 14]. On the other hand, it has to be compatible with the FHE operations. The EtM approach first applies FHE encryption on the plaintext and only then generates a HMAC for the resulting ciphertext. While in this case, the HMAC no longer has to provide strong security guarantees, it still needs to be compatible with the homomorphic ciphertext operations. Finally, for MtE, the HMAC is created for and attached to the plaintext. This means that unlike for EaM, the FHE scheme now operates on the HMAC-augmented message as one entity. Encapsulating the HMAC as part of the message within the encryption further removes the need for the HMAC to support ciphertext maintenance operations itself. Instead, matters like relinearization, key switching or bootstrapping are taken care of by the FHE scheme.

Zero-Knowledge Proofs (ZKPs) make it possible to prove the correctness of a mathematical statement to a verifying party without having to disclose sensitive inputs. Usually, this is achieved by storing intermediate results of ciphertext-based computations. Oftentimes, they are combined with SNARKs to establish proof systems that attest to the validity of an outcome in a non-interactive and efficient manner. The main difficulty in integrating these *zero-knowledge SNARKs (zkSNARKs)* in FHE protocols lies in the incompatibility of the arithmetic operations, which we already mentioned above. Ciphertext maintenance and non-native ring operations such as rounding are of special concern [14, 60].

Trusted Execution Environments (TEEs) take an entirely different approach: Instead of attempting to establish integrity on a software level, special hardware is trusted with performing private computations. Code is executed in these environments in so-called enclaves which are set up by users and verified to be faithfully created and equipped with trusted code free of vulnerabilities [52]. If these conditions are met, enclaves guarantee that the code cannot be altered by other processes running on the machine. Moreover, TEEs can attest that outputs were generated by correctly executed codes. Since TEEs are often restricted in memory and computing power, working with large FHE ciphertexts and evaluation keys might pose an obstacle. Moreover, by combining TEEs and FHEs, the confidentiality guarantees become redundant and cause unnecessary overheads, since data undergoes two layers of encryption. First, plaintext is turned into ciphertext by encrypting it with the FHE scheme. Second, all internal TEE information, including these ciphertexts, gets encrypted in memory [57]. It is also noteworthy that the threat model of TEEs is different since they shift the problem of trust to certificate agencies and hardware manufacturers [14].

Recently, a new line of work introduced *Algorithm-Based Fault Tolerance (ABFT)* as a technique to add verifiability to FHE schemes [57]. For ABFTs, checksums are deployed to protect against unintentional errors or hardware failures. While these systems cannot ensure verifiability against malicious adversaries on their own, this can be achieved by

integrating homomorphic encryption. Note that ABFTs mostly work with matrices so the question arises, how to efficiently extend them to other data types and non-linear operations.

The main result of this thesis is the construction of verifiable PIR from verifiable FHE as presented in Section 6. To allow proper reasoning about how to integrate verifiability, it is necessary to settle on a concrete method that establishes this add-on property. The classification above suggests the use of either homomorphic MACs, ZKPs respectively zkSNARKs, TEEs or ABFTs. We ruled out ABFTs since they seem to be too understudied in our context and are, as of now, mostly used with matrices, while we want to work with vectors. Regarding TEEs, we disliked shifting the trust assumptions from the software to the hardware level and thereby avoiding to address the actual problem of equipping clients with sufficient tools to protect themselves from malicious parties. From the remaining two, zkSNARKs and HMACs, we preferred the first approach for two main reasons. On the one hand, unlike HMACs, it supports two-party computation [60, 43]. This setting is required, since clients and servers alike contribute data to the vPIR protocol. On the other hand, SNARKs are designed to be efficient and are capable of providing security guarantees that are among the strongest currently known for vFHE [50, 43]. Therefore, we opted for using SNARKs in our construction of vPIR. Indeed, we discovered that it already suffices to consider SNARGs and their corresponding security assurances. More details on the concrete integration of SNARGs into FHE protocols and how their properties are utilized are given in Section 6.

4.3 Related Work on vFHE

Like for vPIR, we exemplarily sketch some existing approaches for introducing verifiability to FHE schemes in Table 4. Since there has already been much research into this topic, our selection is rather exclusive and subject to what appears of interest to us. Similarly to the overview table on vPIR (see Table 2), we present the main ideas behind each scheme, mention the assumptions the protocols are based on as well as the category of classification they best belong to according to the above taxonomy. Note that there are schemes that fall under none of the four categories directly, in which case we try to classify it ourselves as best as possible. Additionally, we list their security guarantee, state in how far schemes achieve the compactness condition and give some further interesting background information. In case a scheme does not come with a concrete name, we refer to it by its first author.
The order of the table entries is based on the security notion as first sorting criterion. Starting with schemes that achieve IND-CPA security, we move to the stronger notions of IV-CCA and finally vCCA. This transition is, like before, mirrored in changing from dotted to solid lines separating the entries. Within each category, we arranged the works according to their assumptions. Hereby, protocols requiring weaker assumptions are listed first. Note that the IND-CPA-secure schemes are further divided into those utilizing SNARKs and those that do not. Due to their different approaches, schemes belonging to the last group are somewhat difficult to compare. Moreover, it is a general issue that not all assumptions of a work may be transparent. Hence, we ordered the entries to the best of our knowledge and do not claim absolute certainty.

Some of the works presented already feature tables for scheme comparisons themselves; we refer to [60] and [64] as two examples. Their main insights are, that so far, only very few schemes support ciphertext maintenance operations and security that is stronger than IND-CPA. While TEE-based approaches seem to be the most flexible, integrity arising from ZKPs also offers potential for vFHE constructions and their extensions. Only HMACs come, as of now, with limited options and scopes of security. So far, the highest level of security is achieved by utilizing SNARKs. However, schemes that are solely based on the LWE assumption can give strong security guarantees, too. It is noteworthy, that stronger security guarantees might compromise compactness.

Table 4: Overview of existing vFHE schemes.

Scheme	Main Ideas	Assumptions & Classification	Security Level	Compactness	Miscellaneous
VERITAS [14]	• replicas and precomputable challenge values inserted into plaintext vector • encode messages as lines defined by their evaluation at zero (out: message) and at a secret scalar (out: challenge value) • verification: Agreement of redundancies? Precomputed challenges = received values?	• (R)LWE • category: HMAC	• IND-CPA	compact	• all operations supported • costly preprocessing: amortized complexity • context-dependent large communication or computation overheads • aborts leak information • precomputation fixes the evaluated function in advance
Gennaro et al. [29]	• Yao's garbled circuit construction [65] is one-time secure • combined with FHE by encrypting the input circuit labels	• existence of one-way functions • secure FHEs • category: Non-interactive proofs	• IND-CPA secure • vulnerable to key-recovery attacks	non-compact: ciphertexts grow linearly in $\lvert\mathsf{Eval}\rvert$	• one-time, yet expensive preprocessing stage • efficiency relies on amortized costs • supports arbitrary functions • dynamic and adaptive choice of function inputs throughout the protocol • restarts preprocessing after aborts
DataSeal [57]	• plaintext encoding: added redundancy via secret-dependent, weighted checksums • two-fold verification: correct outcome, manipulation-free data after processing • FHE encryption after encoding	• RLWE • pseudorandom functions • category: ABFT	• IND-CPA	compact	• low space overhead • scalable computation overhead decreasing in the matrix size • efficiency decrease for non-matrix data types • re-encoding for every operation

continued on the next page

Table 4 – *continued from the previous page*

Scheme	Main Ideas	Assumptions & Classification	Security Level	Compactness	Miscellaneous
CHEX-MIX [52]	• service provided within TEE enclave • secure communication channels to enclave • certificates for correct enclave instantiation and channel setup • HE computation initiated by encrypted client input	• (R)LWE • circular-secure HE scheme • trusted Certification Agency • security of TLS protocol • identification of malformed enclaves by processing units • category: TEE	• IND-CPA • integrity against rational adversaries (bounded by incentive to provide somewhat meaningful services)	compact	• one-time setup of secure communication channels required • after one-time initialization phase, actual server is replaced by enclave • client relieved of verifying the correctness of its code to the TEE
Bois et al. [10]	• homomorphic hashing of input and output ciphertexts to reduce the ciphertext space • SNARG proves the hashed, encrypted response to equal the evaluation of the hashed, encrypted inputs	• somewhat HE • homomorphic hash functions • SNARG (for deterministic polynomial-time computations) • random oracle model (for non-interactivity) • category: SNARG	• IND-CPA	compact	• supports all functions of bounded multiplicative depth • adaptable to support non-deterministic computations • public delegation and verification • multi-function • trusted setup required
Viand et al. [60, 43]	• integrate zk-SNARK into FHE • Eval produces a correctness proof for the ciphertext • Vf is the standard zk-SNARK verification	• IND-CPA secure FHE • zero-knowledge SNARKs • category: SNARK	• claimed: IND-CCA1 • reality: IND-CPA [50]	compact	• public delegation • public verifiability

continued on the next page

Table 4 – *continued from the previous page*

Scheme	Main Ideas	Assumptions & Classification	Security Level	Compactness	Miscellaneous
Phalanx [69]	• SNARK-then-FHE: SNARK proofs for encrypted data operate on plaintext level; usual FHE pipeline • formulate circuit as R1CS relation provable by SNARKs • commitment to server input • reduce correctness proofs to random queries to polynomials	• multi-linear polynomial commitment – error correcting codes – Merkle-tree hash • polynomial interactive oracle proof • SNARKs from bilinear maps • category: SNARK	• IND-CPA	compact	• supports packed ciphertexts • commitments rely on natural operations (matrix mult., scalar product) • notion of knowledge soundness: prevents client-input dependent attacks; server must know input in plaintext • efficient: SNARK-then-FHE supports typical arithmetic operations
Yang et al. [64]	• Naor-Yung double encryption paradigm: Encrypt one message under two different pks + NIZK proof that plaintexts agree • index batch arguments and predicate extractable commitment for general predicates proving validity of evaluated ciphertexts	• collision resistant hash • circular-secure LWE • category: ZKP	• IV-CCA security	non-compact [15]: ciphertext size polylogarithmic in |Eval|	• single-hop: evaluated ciphertexts cannot be input into Eval
Checri et al. [15]	• two-ciphertext paradigm [9] applied to Paillier encryption scheme [53] • two ciphertexts produced with different randomness; agree (up to randomness) modulo the message space size	• LOH • category: Non-falsifiable assumption + two-ciphertext paradigm	• vCCA	compact	• only linear HE schemes supported • difficult to generalize

continued on the next page

Table 4 – *continued from the previous page*

Scheme	Main Ideas	Assumptions & Classification	Security Level	Compactness	Miscellaneous
Manulis et al. [50]	• embedding FHE into IND-CCA2 secure encryption scheme • ciphertexts (cts) divided into homomorphic and CCA2 part • Eval operates on homomorphic ct part; outputs are invalid cts in the enclosing CCA2 scheme • SNARK proof for relating evaluated ct to its inputs	• black-box weak SNARKs with simulation-sound extractability • IND-CCA2-secure encryption scheme • category: SNARK	• vCCA	compact (public verifiability), ct size linear in the number of Eval inputs (designated-verifier)	• first IND-CCA1-secure FHE scheme based on bootstrapping • variant for IND-CCA1-secure schemes with relaxed assumptions • variants for symmetric and asymmetric encryption schemes • variants for designated and public verifiability

5 Revisiting a Generic PIR-from-FHE Construction

Among the main contributions of this thesis is a generic construction that starts with verifiable FHE schemes and yields verifiable PIR. Our main source of inspiration is a similar construction from ordinary FHE to PIR in the semi-honest setting [66]. In the following, we outline their main ideas that assume single-bit database entries and propose variants thereof for vectors and matrices. For the latter part of this thesis, we are primarily interested in the vector formulation because it is more efficient than working over single bits and many (v)FHE schemes support vector operations.

5.1 Single-bit Operations

We begin with presenting the original construction of PIRs from FHE schemes [66]. Let $DB = (DB_1, \ldots, DB_n)$ be a database of length n with binary entries $DB_i \in \{0,1\}$ for all $i \in [n]$. Let $\mathcal{I}(DB)$ be its index set, i.e., $\mathcal{I}(DB) = [n]$. Write $(\beta_{j,1}, \ldots, \beta_{j,\ell})$ for the binary representation of some index $j \in \mathcal{I}(DB)$, where $\ell \coloneqq \lceil \log(n) \rceil$. Furthermore, let $\mathsf{FHE} = (\mathsf{FHE.KGen}, \mathsf{FHE.Enc}, \mathsf{FHE.Eval}, \mathsf{FHE.Dec})$ be an IND-CPA-secure FHE scheme.
To query an index $i \in \mathcal{I}(DB)$, represent it in binary $(\alpha_1, \ldots, \alpha_\ell)$. Define the function

$$f(i, \mathcal{I}(DB)) \coloneqq \sum_{j \in [n]: DB_j = 1} \Big(\prod_{k=1}^{\ell} (\alpha_k + \beta_{j,k} + 1) \Big), \tag{1}$$

where addition and multiplication are over $\mathbb{F}_2$. This function f realizes the PIR functionality on plaintexts. That is, f corresponds to the computations the server performs on plaintext level to generate its response.
To transfer f to the level of ciphertexts, we first need some additional notation. For any bit $x \in \{0,1\}$, we write $\widehat{x}$ for its encryption under the FHE scheme, that is

$$\widehat{x} \coloneqq \mathsf{FHE.Enc}_{\mathsf{pk}}(x) \text{ for all } x \in \{0,1\}.$$

Further, let $\widehat{i}$ be the component-wise encryption of the binary representations of i. Then, set

$$\widehat{\mathcal{I}}(DB) \coloneqq \Big\{ \widehat{j} \,|\, j \in \mathcal{I}(DB) \Big\}.$$

Represent the homomorphic ciphertext addition and multiplication by $\oplus$ and $\otimes$, respectively. We are now set up to define the PIR functionality f over ciphertexts, which is denoted by $\widehat{f}$.

$$\widehat{f}(\widehat{i}, \widehat{\mathcal{I}}(DB)) \coloneqq \mathsf{FHE.Eval}_{\mathsf{pk}}(f, \widehat{i}, \widehat{\mathcal{I}}(DB)) = \bigoplus_{j \in [n]: DB_j = 1} \Big(\bigotimes_{k=1}^{\ell} (\widehat{\alpha}_k \oplus \widehat{\beta}_{j,k} \oplus \widehat{1}) \Big).$$

The concrete PIR protocol then takes the following form.

- **Setup**: Let λ be the security parameter. The client generates the key pair $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{FHE.KGen}(1^\lambda)$, sends the public key pk for encryption and evaluation to the server and keeps the secret key sk for decryption to itself.

  The server, holding a database DB as described above, precomputes the binary representations of its indices $j \in \mathcal{I}(DB)$ and encrypts them with the public key pk to obtain $\widehat{\mathcal{I}}(DB)$.

- **Query**: Assume that the client wants to retrieve the i^{th} database file for some $i \in [n]$. It first represents the index in binary, yielding $(\alpha_1, \ldots, \alpha_\ell)$, and then encrypts every entry with the public key pk. The resulting vector $\widehat{i} = (\widehat{\alpha}_1, \ldots, \widehat{\alpha}_\ell)$ is sent to the server as query.
- **Response**: The server receives the encrypted query index $\widehat{i}$ and homomorphically computes $\widehat{r} \coloneqq \widehat{f}(\widehat{i}, \widehat{DB}, \widehat{\mathcal{I}}(DB))$. It then outputs $\widehat{r}$ as response.
- **File extraction**: To obtain the content of the desired database entry, the client merely has to decrypt $\widehat{r}$ using its secret key. The result is then equal to $DB_i \in \{0, 1\}$.

We prove the correctness of the scheme described above. Note that is suffices to argue on the level of plaintexts, since then, correctness of the underlying FHE scheme guarantees the correctness for ciphertexts. We begin with a helpful observation.

Lemma 5.1. *Let $\gamma_j \coloneqq \prod_{k=1}^{\ell}(\alpha_k + \beta_{j,k} + 1)$ for all $j \in [n]$, where $(\alpha_1, \ldots, \alpha_\ell)$ and $(\beta_{j,1}, \ldots, \beta_{j,\ell})$ are the binary representations of the query index i and the indices of the database $j \in \mathcal{I}(DB)$, respectively. Then,*

$$\begin{aligned} \gamma_j = 1 &\iff (\alpha_1, \ldots, \alpha_\ell) = (\beta_{j,1}, \ldots, \beta_{j,\ell}) \\ &\iff i = j. \end{aligned}$$

Proof. Let $j \in [n]$ be some database index. Observe that the second bi-implication is a direct consequence of $(\alpha_1, \ldots, \alpha_\ell)$ and $(\beta_{j,1}, \ldots, \beta_{j,\ell})$ being the binary representations of i and j, respectively. Therefore, we are left with proving $\gamma_j = 1 \iff i = j$.
First, assume $j \neq i$. This means that the binary representations of i and j differ in some position $t \in [\ell]$. Hence, at this position, exactly one of α_t and $\beta_{j,t}$ takes the value one; the other is zero. Thus,

$$\alpha_t + \beta_{j,t} + 1 = 0$$

and further

$$\gamma_j = (\alpha_t + \beta_{j,t} + 1) \cdot \prod_{\substack{k=1 \\ k \neq t}}^{\ell} (\alpha_k + \beta_{j,k} + 1) = 0 \cdot \prod_{\substack{k=1 \\ k \neq t}}^{\ell} (\alpha_k + \beta_{j,k} + 1) = 0.$$

For the second direction, assume $\gamma_j = 0$. Then, at least one factor of the product must be zero. This in turn implies that for at least one $t \in [\ell]$, we have $\alpha_t \neq \beta_{j,t}$. Consequently, $i \neq j$, which concludes the proof. □

With this lemma at hand, correctness of the above PIR protocol follows easily: The response $r \coloneqq f(i, \mathcal{I}(DB))$ is computed as the sum over all γ_j for which the j^{th} database entry DB_j is one. That is

$$r = f(i, \mathcal{I}(DB)) = \sum_{j \in [n] : DB_j = 1} \gamma_j.$$

By Lemma 5.1, we know that γ_j is always zero except if $j = i$. Thus, in case $DB_i = 0$, γ_i does not appear as a summand of r, meaning that r is just the zero sum. If, however, $DB_i = 1$, γ_i constitutes the value one to the output r. Moreover, it is the only non-zero

summand of r. In summary, for honest computations, we yield the result $r = DB_i$.

This construction has several shortcomings. First, since it is based on the encryption and querying of individual bits (database files are single bits), it is costly and does not scale well with the size of the database. Second, it relies on the non-arithmetic selection of the subset of $\mathcal{I}(DB)$ comprising of all indices j such that $DB_j = 1$. Third, it falls short of treating all parameters in a uniform manner: While the client's query and the database indices get encrypted, the response functionality $\widehat{f}$ still works over the database entries in plaintext. In order to improve the construction, we contemplated the following aspects. Depending on the underlying FHE scheme, plaintext and ciphertext space might not agree, making it difficult to operate on both simultaneously. Moreover, the concrete verification method in use might require the generation of certain verification data alongside the ciphertexts [60, 14, 30, 71]. Thus, for verification to work, it might be necessary to process the database further and not to rely on its plaintext content for the PIR functionality. To overcome these hindrances, we propose reformulations of the above construction for matrices and vectors alike.

5.2 Matrix Operations

Starting with the construction variant for matrices, we remark that the formulation we derive in the following is not very practical. It is primarily introduced to motivate the subsequent construction using vector operations, which serves as a main tool for the remainder of this thesis. Therefore, in this subsection we mainly aim for a comprehensive derivation of the matrix formulation and leave rigorous proofs to Subsection 5.3 concerning the vector variant.
To turn the above construction for single-bit data into matrix representation, we leverage the following linearity property of diagonal matrices with respect to multiplication. Let $m, n \in \mathbb{N}$ and let

$$\begin{pmatrix} a_{11}^i & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & a_{nn}^i \end{pmatrix}$$

be any $n \times n$-matrix for all $i \in [m]$. Here, we do not specify the diagonal entries $a_{j,j}^i$ further. For simplicity, they can be pictured to belong to any ring of the reader's liking. Then,

$$\begin{pmatrix} \prod_{i=1}^m a_{11}^i & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \prod_{i=1}^m a_{nn}^i \end{pmatrix} = \prod_{i=1}^m \begin{pmatrix} a_{11}^i & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & a_{nn}^i \end{pmatrix}, \forall m, n \in \mathbb{N}. \tag{2}$$

We sketch the strategy for writing the PIR functionality f in Equation (1) in terms of matrices. From now on, consider a database

$$DB = \{DB_{j,s}\}_{\substack{j=1,\dots,n \\ s=1,\dots,m}} \in \{0,1\}^{n \times m}$$

that consists of n files, each of length m. Instead of computing the product terms γ_j defined in Lemma 5.1 for some subset of the index set $\mathcal{I}(DB)$, they are now calculated

for all $j \in \mathcal{I}(DB)$ and constitute the entries of a diagonal $n \times n$-matrix

$$\begin{pmatrix} \gamma_1 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \gamma_n \end{pmatrix} = \begin{pmatrix} \prod_{k=1}^{\ell}(\alpha_k + \beta_{1,k} + 1) & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \prod_{k=1}^{\ell}(\alpha_k + \beta_{n,k} + 1) \end{pmatrix}.$$

From this representation, we derive the format the client and server inputs have to take. To this end, mentally apply the linearity of Equation (2) to expand the ℓ-fold products on the diagonal, i.e., $\prod_{k=1}^{\ell}(\alpha_k + \beta_{j,k} + 1)$, into a product of ℓ separate matrices. Furthermore, each of these can be split into three matrices by using matrix linearity for addition. That is, for all $k \in [\ell]$

$$\begin{aligned} \begin{pmatrix} \alpha_k + \beta_{1,k} + 1 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \alpha_k + \beta_{n,k} + 1 \end{pmatrix} &= \underbrace{\begin{pmatrix} \alpha_k & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \alpha_k \end{pmatrix}}_{:= A_k} + \underbrace{\begin{pmatrix} \beta_{1,k} & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \beta_{n,k} \end{pmatrix}}_{:= B_k} + \underbrace{\begin{pmatrix} 1 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & 1 \end{pmatrix}}_{:= \mathbf{1}} \\ &= A_k + B_k + \mathbf{1}. \end{aligned} \tag{3}$$

Observe that for all $k \in [\ell]$, A_k and B_k consist purely of client and server data, respectively, and can therefore serve as the respective party's input. We set $\mathbf{1}$ as the notation of the identity matrix of size $n \times n$.

Next we motivate how to arrive at the final response r, again starting from the matrix with γ_j for $j = 1, \dots, n$ on the diagonal. Recall that by Lemma 5.1, all γ_j for $i \neq j \in [n]$ are simply zero. Thus, when multiplying the diagonal matrix with the database from the right, the outcome is a matrix that consists of a single non-zero row, which is precisely the requested database file. The server can add up all rows and send the result as a response to the client. This even reduces communication costs, since now, the final answer is a vector, not a matrix.

We translate this strategy into a precise PIR protocol. Let λ be the security parameter, $DB \in \{0,1\}^{n \times m}$ be a database in matrix form with index set $\mathcal{I}(DB)$.

- **Setup**: Like before, the client generates the key pair $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{FHE.KGen}(1^\lambda)$, publishes the public key pk and privately stores the secret key sk.

  The server precomputes the matrices B_k for $k \in [\ell]$ from the index set $\mathcal{I}(DB)$ and encrypts them with the public key, yielding $\left\{\widehat{B}_k\right\}_{k=1,\dots,\ell}$.

- **Query**: Let $i \in [n]$ be the query index. From the binary representation of i, the client constructs the matrices A_k for all $k \in [n]$ and encrypts each with the public key pk. The query is the set $\left\{\widehat{A}_k\right\}_{k=1,\dots,\ell}$.

- **Response**: Let $\widehat{\mathbf{1}}$ be an encryption of the identity matrix of size $n \times n$. Taking inspiration from Equation (3), for all $k \in [\ell]$, the server sets

$$\widehat{C}_k := \widehat{A}_k \oplus \widehat{B}_k \oplus \widehat{\mathbf{1}},$$
$$\widehat{C} := \bigotimes_{k=1}^{\ell} \widehat{C}_k.$$

Furthermore, define the $n \times n$-matrix

$$\text{AddCol} \coloneqq \begin{pmatrix} 1 & 1 & \dots & 1 \\ 0 & 0 & \dots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \dots & 0 \end{pmatrix},$$

consisting of ones in the first row and zeros elsewhere. The name is motivated by the effect is has when multiplying it to matrices from the left: Every slot of the resulting first row is the summation of its corresponding matrix column. All other rows are zero. The server uses it to homomorphically compute

$$\widehat{R} \coloneqq \widehat{\text{AddCol}} \otimes \widehat{C} \otimes \widehat{DB} \tag{4}$$

and sends the first row of $\widehat{R}$ to the client.

- **File extraction**: Upon receiving the first row of $\widehat{R}$, the client enlarges it again to a matrix, decrypts it with its secret key and retrieves the first row. As motivated by the above description of the strategy, if all computations are carried out correctly, the result is the i^{th} file of the database DB.

Remark 5.2. There are two technicalities we want to point out.

- **Keeping matrix form** We remark, that the additional zero-rows of the matrix AddCol only serve the purpose of keeping to the matrix form. The resulting matrix $\widehat{R}$ carries relevant information in the first row alone. Theoretically, the server could reply to the client's request with all of $\widehat{R}$, without extracting the first row. However, the size of the response would then be $n \times m$ and thus as large as the database. This would violate our incentive of running a PIR instance in the first place. The requirement of keeping to a specific matrix form also necessitates the client-side workaround of reconstructing a matrix from the given response before decryption. Since this artificial expansion adds no further information, the client can again neglect all but the first row afterwards.

- **Different-sized databases** To be entirely accurate, for $m > n$, one would need to split the database into matrices of size $n \times n$ each and carry out the response computation for every database part separately. Similarly, for $m < n$, the database would need to be padded with additional $n - m$ zero columns. Both is required because encryption schemes usually work over plaintexts of the same size. Since we mainly introduced the matrix formulation of the PIR construction to make the transition to the vector variant more comprehensible, we refrain from detailing this aspect here. In fact, it appears again when working over vectors, so we simply postpone the discussion.

5.3 Vector Operations

Our final vector-based PIR-from-FHE construction is motivated by two aspects: First, matrices and especially their ciphertexts are typically large, making them impractical to

handle. Second, it is more common for FHE schemes to support vector instead of matrix operations (e.g. [14, 40, 10] or, most prominently, the BGV [11] and BFV [26] schemes). The formulation for vectors readily follows the blueprint for matrices. For the conversion, it mostly suffices to aggregate the diagonal entries of the matrices to form vectors. Only the final computation of $\widehat{R}$ needs adaptation. We show how to realize it using component-wise homomorphic addition, multiplication and rotation alone, which are standard operations supported by most FHE schemes. We remark that while the inputs to both the PIR and the FHE scheme are now vectors, the database remains shaped as a matrix.
To start, for all $k \in [\ell]$, define

$$a_k \coloneqq (\alpha_k, \dots, \alpha_k),$$
$$b_k \coloneqq (\beta_{1,k}, \dots, \beta_{n,k}),$$

and let $\mathbf{1}$ now denote the all-one vector of length n. Their respective FHE encryptions are again indicated by the $\widehat{\cdot}$ symbol. Like for matrices, let

$$\widehat{c}_k \coloneqq \widehat{a}_k \oplus \widehat{b}_k \oplus \widehat{\mathbf{1}},$$
$$\widehat{c} \coloneqq \bigotimes_{k=1}^{\ell} \widehat{c}_k.$$

The database $DB \in \{0,1\}^{n \times m}$ is encrypted column-wise, leading to m encrypted vectors $\left\{\widehat{DB}_{\cdot,s}\right\}_{s=1,\dots,m}$.
We now describe how to adapt the response computation of Equation (4) for vectors, using vector addition, multiplication and rotation alone. Since the reasoning heavily relies on the exact positioning of certain values within the vectors, we explain the computation on the plaintext level. It carries over to ciphertexts by replacing the operations with their homomorphic analogues. We remark that position counting starts at one. That is, we consider the first component of a vector to be at position one.
Let c be the underlying plaintext of the encryption $\widehat{c}$ and let $e_j \coloneqq (0, \dots, 0, 1, 0, \dots, 0)$ be the unit vector with 1 at position j, for all $j \in [n]$. Furthermore, $\cdot$ denotes the component-wise multiplication of two vectors and, in particular, results again in a vector. Computing the response can then be defined by the steps listed in Procedure 1. For a better understanding, we also provide an illustration of the procedure in Figure 9 and explain all steps in more detail when proving the correctness of the vector-based PIR-from-FHE construction.
Putting it all together, we obtain the following PIR protocol, only slightly adjusting the one for matrices. Let λ, $DB \in \{0,1\}^{n \times m}$ and $\mathcal{I}(DB)$ be as usual.

- **Setup**: The client generates the key pair $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{FHE.KGen}(1^\lambda)$, publishes the public key pk and privately stores the secret key sk.

  The server precomputes the vectors b_k for $k \in [\ell]$ from the index set $\mathcal{I}(DB)$ and encrypts them with the public key, yielding $\left\{\widehat{b}_k\right\}_{k=1,\dots,\ell}$. Additionally, it generates encryptions of the unit vectors $\widehat{e}_j$ for all $j \in [n]$.

- **Query**: Let $i \in [n]$ be the query index. From the binary representation of i, the client constructs the vectors a_k for all $k \in [n]$ and encrypts each with the public key pk. The query is the set $\{\widehat{a}_k\}_{k=1,\dots,\ell}$.

Procedure 1 Vector Response Computation for Plaintexts

Input: c, $\{e_j\}_{j=1,\dots,n}, \{DB_{\cdot,s}\}_{s=1,\dots,m}$

Output: Set r containing vectors that correspond to the batches making up the encrypted i^{th} database file.

Extract entries:

Set $c_s \coloneqq c \cdot DB_{\cdot,s}$ for all $s \in [m]$.

Compute $e_j \cdot c_s = (0, \dots, c_{j,s}, \dots, 0)$ for all $j \in [n]$, with $c_{j,s}$ as the j^{th} entry of the vector c_s.

Rotate:

Rotate $(0, \dots, c_{j,s}, \dots, 0)$ for all $s \in [m]$ and all $j \in [n]$ so that $c_{j,s}$ is at position

$$\begin{cases} s \mod n & \text{if } s \not\equiv 0 \mod n, \\ s = n & \text{otherwise.} \end{cases}$$

Add within columns:

Add vectors originating from the same column. That is, for all $s \in [m]$, sum up

$$(0, \dots, c_{1,s}, \dots, 0) + \cdots + (0, \dots, c_{n,s}, \dots, 0) = (0, \dots, \sum_{j=1}^{n} c_{j,s}, \dots, 0).$$

Add across columns:

Arrange the column-sum vectors from line 7 in $\left\lceil \frac{m}{n} \right\rceil$ batches of length n.

For each batch, add vectors across columns. That is, for all $d \in [\left\lceil \frac{m}{n} \right\rceil]$ and all $s \in \{(d-1)n+1, \dots, dn\}$ compute

$$\sum_{s=(d-1)n+1}^{dn} (0, \dots, \sum_{j=1}^{n} c_{j,s}, \dots, 0) = (\sum_{j=1}^{n} c_{j,(d-1)n+1}, \dots, \sum_{j=1}^{n} c_{j,dn}).$$

Return $r \coloneqq \left\{ \left(\sum_{j=1}^{n} c_{j,(d-1)n+1}, \dots, \sum_{j=1}^{n} c_{j,dn} \right) \right\}_{d=1,\dots,\left\lceil \frac{m}{n} \right\rceil}$

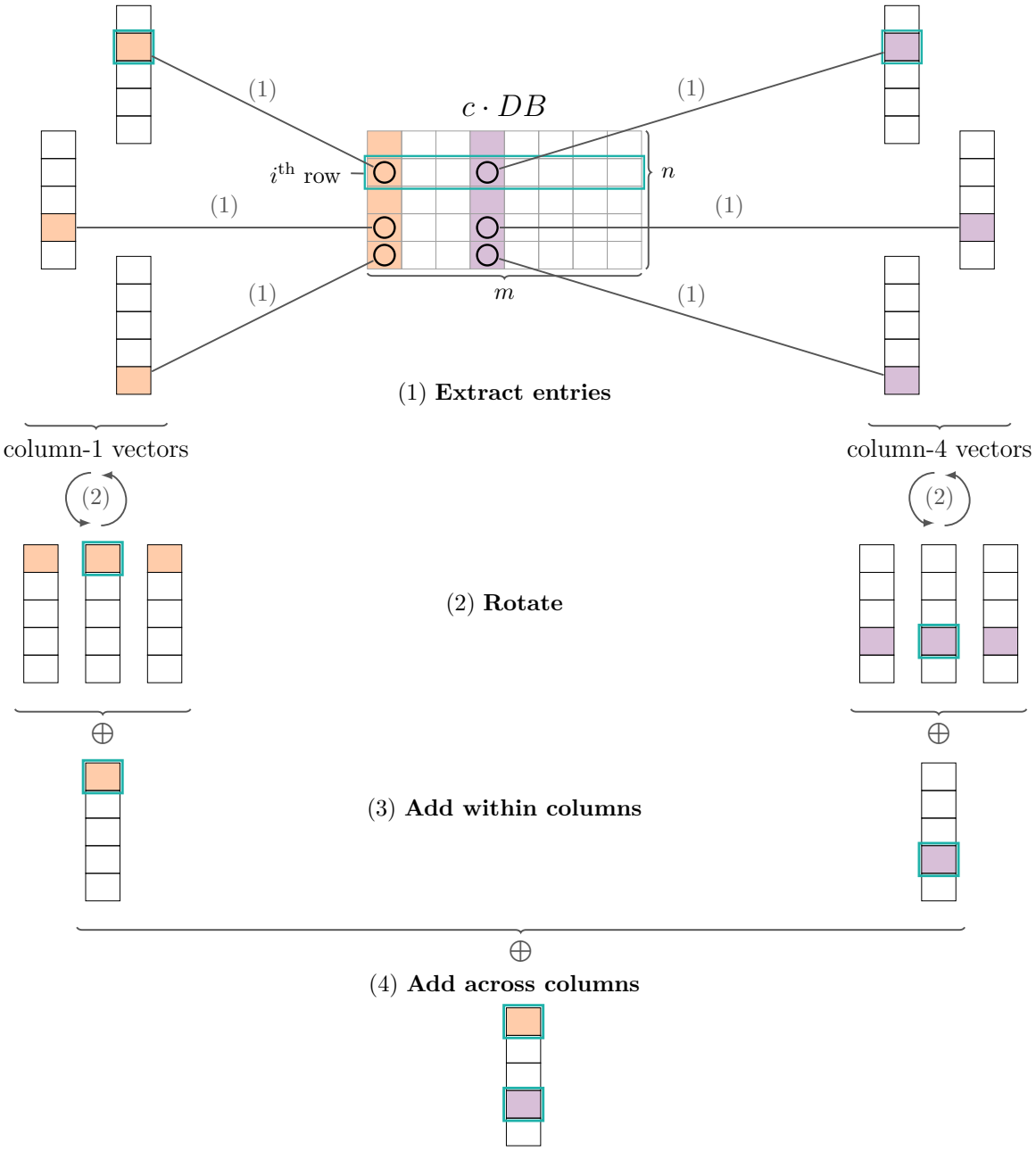

Figure 9: Illustration of the vector response computation of Procedure 1 on plaintext level. Framed cells in blue indicate relevant information on the queried file of index i.

- **Response**: The server sets

$$\widehat{c}_k \coloneqq \widehat{a}_k \oplus \widehat{b}_k \oplus \widehat{\mathbf{1}},$$
$$\widehat{c} \coloneqq \bigotimes_{k=1}^{\ell} \widehat{c}_k.$$

  It performs the homomorphic analogues of the computation steps listed in Procedure 1 over the ciphertext inputs $\widehat{c}, \{\widehat{e}_j\}_{j=1,\dots,n}$ and $\left\{\widehat{DB}_{\cdot,s}\right\}_{s=1,\dots,m}$. The output is defined as $\widehat{r}$ and sent to the client.

- **File extraction**: The client decrypts all vectors contained in the set $\widehat{r}$ separately. By assembling them according to the order of the batches, the client yields the requested file $(DB_{i,1}, \dots, DB_{i,m})$.

Lemma 5.3. *The above vector variant of the PIR-from-FHE construction is correct.*

Proof. Let all parameters be as in the description of the PIR protocol. We have to show that $\mathsf{Dec}_{\mathsf{sk}}(\widehat{r}) = \{DB_{i,s}\}_{s=1,\dots,m}$, from which the client can reconstruct the desired file $(DB_{i,1}, \dots, DB_{i,m})$.
Observe that by construction, the vector $\widehat{c}$ encrypts $(\gamma_1, \dots, \gamma_n)$, where γ_j is the product defined in Lemma 5.1 for all $j \in [n]$. In particular, the lemma states that all but the i^{th} entry of $\widehat{c}$ are encryptions of zero and that the i^{th} entry itself is an encryption of one. Therefore, homomorphically multiplying it with each encrypted database column $\widehat{DB}_{\cdot,s}$ for $s \in [m]$, similarly to line 2 of Procedure 1, yields vectors that likewise encrypt zero at all but the i^{th} position. In these slots we now get encryptions of the i^{th} database bit of the respective column, i.e., encryptions of $DB_{i,s}$ for all $s \in [m]$. This collection of vectors is depicted as the matrix $c \cdot DB$ in Figure 9 and serves as the starting point of the server's response calculation. We again emphasize that here, the operator $\cdot$ refers to the component-wise multiplication of two vectors.
Procedure 1 next singles out every individual database entry by multiplying the database columns with all unit vectors of length n successively (see step (1) in Figure 9). By rotating the resulting vectors according to their column index in step (2), all entries of the same column are aligned. Adding them up as noted in line 7 of the procedure or as depicted at stage (3) of Figure 9, merges all vectors originating from the same column $s \in [m]$ into one that, too, mostly contains encryptions of zero, except for the encryption of the column sum at position $s \mod n$ (or n if $s \equiv 0 \mod n$). The above considerations on the column values show, that the column sums are indeed merely encryptions of the i^{th} database entries, that is

$$\bigoplus_{j=1}^{n} \widehat{c}_{j,s} = \widehat{DB}_{i,s} \text{ for all } s \in [m]. \tag{5}$$

Rotating served the purpose of arranging the column sums so that their vertical order matches the one given by the column indices. Combined with processing the column sum vectors computed in line 7 in batches, it is ensured that no two encryptions of database entries overlap when adding up the vectors as described in line 10 and visualized in step (4). Hence, the vector $(\bigoplus_{j=1}^{n} \widehat{c}_{j,1}, \dots, \bigoplus_{j=1}^{n} \widehat{c}_{j,n})$ resulting from the first batch ($d = 1$) for

example, consists of the encrypted column sum of the first database column $DB_{\cdot,1}$ at the first position, the encrypted column sum of the second database column $DB_{\cdot,2}$ at the second position and so on. Batching is required whenever the database has more columns than rows, that is $m > n$, since then there are more column sums than a single vector has slots. We remark that the server has to indicate the batch number a vector of the final response set $\widehat{r}$ corresponds to. Otherwise, the client does not know in which order to reassemble the vectors in $\widehat{r}$ after decryption. Overall, correctness of the PIR protocol now follows from Equation (5), yielding

$$\begin{aligned}\mathsf{Dec}_{\mathsf{sk}}\left(\Big(\bigoplus_{j=1}^{n}\widehat{c}_{j,(d-1)n+1},\ldots,\bigoplus_{j=1}^{n}\widehat{c}_{j,dn}\Big)\right) &= \left(\mathsf{Dec}_{\mathsf{sk}}\Big(\bigoplus_{j=1}^{n}\widehat{c}_{j,(d-1)n+1}\Big),\ldots,\mathsf{Dec}_{\mathsf{sk}}\Big(\bigoplus_{j=1}^{n}\widehat{c}_{j,dn}\Big)\right)\\ &= \Big(\mathsf{Dec}_{\mathsf{sk}}\Big(\widehat{DB}_{i,(d-1)n+1}\Big),\ldots,\mathsf{Dec}_{\mathsf{sk}}\Big(\widehat{DB}_{i,dn}\Big)\Big)\\ &= \Big(DB_{i,(d-1)n+1},\ldots,DB_{i,dn}\Big)\end{aligned}$$

for all $d \in \left\lceil \frac{m}{n} \right\rceil$. □

6 Constructing vPIR from vFHE

Based on the construction of PIR from FHE and especially its vector formulation established in the previous section, we now proceed with integrating verifiability into the underlying FHE scheme and show that the resulting PIR protocol meets the conditions of verifiable PIR as stated in Definitions 3.3 to 3.6. Our construction aims at being as general as possible while giving sufficient insights into the handling of the verification material. As mentioned in Subsection 4.2, inspired by the classification of vFHE schemes, we contemplated using homomorphic MACs or SNARKs for verification. Hereby, we found that argument systems are most suitable for our use case and therefore, from now on, concentrate on them for establishing verifiability. We leave it as an open question to formulate similar constructions when using HMACs or other verification techniques.
Before introducing our new definitions and construction, we settle on and recall some general assumptions: First, all vFHE schemes the construction is applied to are expected to meet the requirements of Definitions 4.3 to 4.8. Second, we again assume that public parameters are known to all parties at all times and can therefore be omitted as explicit input to functions. The same holds for a party's private data and the messages it receives throughout a protocol.

6.1 Definitions

To begin with, we define verifiable FHE schemes that rely on argument systems to attest to the correctness of their evaluation results. For this, we follow the standard approach of employing such systems in the context of FHE [60, 30, 50] but refrain from including redundant information. Notably, we do not equip the output of the encryption function with verification tags, since these are usually set to a copy of the ciphertexts. To avoid misunderstandings, we further fix two interpretations of the general setup.
First, the index set $\mathcal{I}(DB)$ of the database DB is viewed as static with respect to the database files. That is, whichever file is currently at position $i \in [n]$ is referred to as the i^{th} file. Even if this file had previously been at position $j \neq i$ and was only later moved to position i, it takes on the new index i rather than retaining the old one, j.
We view the server as a black box offering some service. Consequently, inputs to the two-party (v)PIR protocol are now solely provided by a single party. More specifically, only the client provides the input and the server, upon receiving it, computes a (v)PIR-compliant response over its database DB. In particular, we consider DB and $\mathcal{I}(DB)$ to be knowledge the server possesses internally. When proving the correctness of the answer, it is used as witness. We propose the notation f_w when f is computed with knowledge w. It is reminiscent of processing knowledge as an additional function input, i.e., $f(w, \cdot)$, as is the case in the alternative interpretation of (v)PIR as a two-party protocol with inputs from both parties. Unless changed deliberately, we assume knowledge to be constant: Apart from the encrypted client query, all inputs to the server are public knowledge. Further, by design, the query reveals no information about the index of interest. Combining these two observations, by running the (v)PIR response algorithm, the server gains no additional knowledge beyond what it could already have known before executing the protocol. This justifies neglecting potential knowledge growth over time. Besides, we assume that w can be revealed to the client at any time and does not need to be kept secret. Altogether, this

assumption streamlines notation and feels natural since one usually imagines querying a database directly rather than sending the query to a server that then inputs its own database into an abstract functionality to compute the response.

6.1.1 Protocol for SNARG-based vFHE

We now propose a definition of verifiable FHE that provides sufficient structure for transforming it into a verifiable PIR protocol. Most importantly, tools are required to bind the computing party to its internal knowledge as this imitates the digest distribution crucial to vPIR protocols in the single-server setting. Hereby, we immediately aim at tolerating maliciously generated commitments and only later show, how to adjust the definition whenever commitments are guaranteed to be honest. Because of these strong security assurances, we refer to the new notion of vFHE as *strongly vFHE*. Its variant for truthfully computed commitments we call *weakly vFHE*.
The main idea is to let the server produce some value from its knowledge w that binds it to using w in all subsequent computations. It shares this value with the client, that can then verify the truthfulness of the server's evaluation results later on. Judging from the current literature, it is common to utilize commitment schemes to establish the desired binding property, especially if tampered knowledge shares are to be considered [69, 25, 13, 55]. This motivates our decision to integrate a commitment scheme into our definition of strongly vFHE. Thereby, we assume deterministic commitments: Committing to a value twice should result in the same commitment. This is justified since we only commit to public values that need not be protected by introducing non-determinism and randomness. It has the additional benefit of simplifying the construction significantly, as changes in the underlying data can be detected by directly comparing digests. Moreover, our purposes require commitment schemes to be capable of coping with malicious behavior. That is, executing the commitment verification and identifying honest commitments have to be feasible for the client. This is essential to ensure that the client holds data that unmistakably relates to the genuine knowledge of the server, i.e., the actual database. Note that here, *feasible* means that running the vPIR protocol including all verification steps, is more efficient than downloading the entire database, which would undermine the PIR incentive. In particular, the client can verify the correctness of the commitment from data that is small relative to the size of the database. We refer to the end of Subsection 2.2 for more information and examples of commitment schemes that meet all these criteria.
The above considerations are reflected in the following definition for strongly verifiable FHE schemes. It is further based on the works of [60] and [50].

Definition 6.1 (Strongly verifiable FHE fit for SNARGs - strongly vFHE)**.** Let $\mathsf{FHE} = (\mathsf{FHE.KGen}, \mathsf{FHE.Enc}, \mathsf{FHE.Eval}, \mathsf{FHE.Dec})$ be an IND-CPA-secure FHE scheme. Further, let $\Pi = (\Pi.\mathsf{VGen}, \Pi.\mathsf{Prove}, \Pi.\mathsf{Vf})$ be a Succinct Non-interactive ARGument system (SNARG) and let f be the public function to be evaluated homomorphically and verifiably. Moreover, let $\mathsf{Com} = (\mathsf{Com.Gen}, \mathsf{Com.Commit}, \mathsf{Com.Vf})$ be a deterministic commitment scheme, capable of detecting malicious digests in a feasible manner. A *strongly verifiable Fully Homomorphic Encryption scheme (strongly vFHE)* is given by a tuple of PPT algorithms $(\mathsf{Gen}, \mathsf{Commit}, \mathsf{VerCommit}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{Vf}, \mathsf{Dec})$ defined by

- $\mathsf{Gen}(1^\lambda, \mathcal{R}_f) \to ((\mathsf{pk}, \mathsf{crs}, \mathsf{ck}), (\mathsf{sk}, \tau, \mathsf{vk}))$: Depending on the security parameter λ, the *generation* algorithm generates the following parameters.

  - Invoke $(\mathsf{pk}, \mathsf{sk}) \coloneqq \mathsf{FHE.KGen}(1^\lambda)$ to produce a key pair for en- and decryption.
  - Set the commitment and the verifier key to $(\mathsf{ck}, \mathsf{vk}) \coloneqq \mathsf{Com.Gen}(1^\lambda)$.
  - For verification, information depending on the relation $\mathcal{R}_f$ is generated, namely a common reference string crs and a verification tag τ. Hereby,

$$\begin{aligned}\mathcal{R}_f \coloneqq \{((f, c_x, c_y, \tau_w), (\tau'_w, w)) \mid\ & c_y = \mathsf{FHE.Eval}_{\mathsf{pk}}(f_w, c_x) \\ & \wedge\ w \text{ is some valid witness} \\ & \wedge\ (\tau'_w, \_) = \mathsf{Com.Commit}_{\mathsf{ck}}(w) \\ & \wedge\ \tau_w = \tau'_w\}.\end{aligned}$$

  - Publish the tuple of all public keys $(\mathsf{pk}, \mathsf{crs}, \mathsf{ck})$, while keeping the secret keys $(\mathsf{sk}, \tau, \mathsf{vk})$ to the verifying party.

- $\mathsf{Commit}_{\mathsf{ck}}(w) \to (\tau_w, \omega_w)$: For w being the internal knowledge of the computing party, compute the *commitment* that binds the party to using w in subsequent computations. Output $(\tau_w, \omega_w) \coloneqq \mathsf{Com.Commit}_{\mathsf{ck}}(w)$.

- $\mathsf{VerCommit}_{\mathsf{vk}}(\tau_w, \omega_w) \to b_{com}$: To *verify the commitment*, invoke the algorithm $b_{com} \coloneqq \mathsf{Com.Vf}_{\mathsf{vk}}(\tau_w, \omega_w)$ on the claimed commitment τ_w and the opening ω_w. The commitment is accepted only if $b_{com} = 1$; otherwise, it is rejected as incorrect.

- $\mathsf{Enc}_{\mathsf{pk}}(x) \to c_x$: The public-key *encryption* algorithm takes a private input x and outputs $c_x \coloneqq \mathsf{FHE.Enc}_{\mathsf{pk}}(x)$.

- $\mathsf{Eval}_{\mathsf{pk}}(\mathsf{crs}, f, c_x, \tau_w) \to (\tau'_w, c_y, \pi_y)$: Assume that the computing party has some internal knowledge w, which might be empty. The client provides its ciphertext c_x together with the commitment τ_w that has previously been issued by the computing party. Then, in the *evaluation*, the function f is homomorphically applied to the ciphertext c_x and correctness of this computation is proven. This is realized by the following steps.

  - $(\tau'_w, \_) \coloneqq \mathsf{Commit}_{\mathsf{ck}}(w)$ is the server's new commitment to its current internal knowledge w.
  - $c_y \coloneqq \mathsf{FHE.Eval}_{\mathsf{pk}}(f, c_x)$ is the encryption of $y \coloneqq f(x)$.
  - $\pi_y \coloneqq \Pi.\mathsf{Prove}(\mathsf{crs}, f, c_x, c_y, \tau_w, \tau'_w)$ proves that $((f, c_x, c_y, \tau_w), (\tau'_w, w)) \in \mathcal{R}_f$.

  Output (τ'_w, c_y, π_y).

- $\mathsf{Vf}_{\mathsf{sk}}(\tau, \tau_w, \tau'_w, c_x, c_y, \pi_y) \to b_{ver}$: First, check if the commitments agree, i.e., if $\tau_w = \tau'_w$. If not, set $b_{ver} = 0$ and return. Otherwise, compute the value of the acceptance bit $b_{ver} \coloneqq \Pi.\mathsf{Vf}_{\mathsf{sk}}(\tau, \tau_w, \tau'_w, c_x, c_y, \pi_y)$. If c_y is found to be truthful, set $b_{ver} = 1$. Otherwise, set $b_{ver} = 0$ and reject the response c_y.

- $\mathsf{Dec}_{\mathsf{sk}}(c_y) \to y$: If verification was successful ($b_{ver} = 1$), the *decryption* algorithm recovers the evaluation result $y = f(x)$ from the ciphertext c_y. That is, the output is $y \coloneqq \mathsf{FHE.Dec}_{\mathsf{sk}}(c_y)$.

Remark 6.2. We highlight three aspects of the above definition.

- **Function-independent key generation** As already mentioned in Remark 4.2, in Gen we separate the key generation of the FHE scheme from that of the SNARG component. In particular, this allows FHE.KGen to be function independent.

- **Relation setup** It is crucial for the security of the verification method to keep the verification tag secret to the client. Consequently, either the client itself or a trusted third party has to execute the setup algorithm. Since we refrain from introducing additional parties and delegating trust assumptions, we rely on the client to be capable of performing the setup. This includes computing a common reference string for the relation $\mathcal{R}_f$, which is based on information from both the client and the computing party. So, does the client need to know the witness values beforehand? In [71], a scenario is studied that is very similar to ours: To construct a verifiable PIR protocol, the authors leverage SNARK proofs to ensure correctness of the server's responses. By making use of quadratic arithmetic circuits, they establish a proof system for a relation that takes data from the client and the server alike and that can be set up by the client alone. This suggests that the client is able of generating the verification data without having to know the witness first, which, in the PIR setting, would be the entire database the server hosts. A more detailed discussion on arithmetic circuits in the context of argument systems presented in [38] supports this claim. While it is necessary that the circuit construction takes the witness into account by means of attributing certain wires to it and by respecting its relation-specific constraints, the concrete values it takes are of no relevance.

- **Number of commitments** Depending on the specification of the commitment scheme used, Com might need to compute the commitments for the database $\{DB_{\cdot,s}\}_s$ and for its encrypted index set vectors $\left\{\widehat{b}_k\right\}_k$ separately. In this case, the server computes one commitment for each piece of data, and sends their collection to the client. To prove membership in $\mathcal{R}_f$, the server recomputes all these commitments and returns them to the client who then compares them to the previous commitments.

Before stating the requirements strongly vFHE schemes have to meet, we outline the interaction between two parties using such a scheme. To handle maliciously generated commitments, we introduce a preprocessing procedure PrepCom that requests a new commitment from the computing party as long as the current commitment is found to be dishonest. That is, at the end of the preprocessing phase, the honest party holds a commitment to the internal knowledge w that passes verification via Com.Vf. As previously mentioned, this step is needed to ensure that the honest party has at least one piece of data that is undoubtedly related to the true internal knowledge w. Otherwise, malicious behavior of the computing party might go undetected. We justify this preprocessing by noting that, unless w changes, it only needs to be performed once for all evaluations to come. Furthermore, since data produced by commitment schemes is typically small, exchanging and verifying commitments is rather efficient.

Remark 6.3. For honest servers, we suggest that they inform clients proactively whenever their internal knowledge is updated. Otherwise, the first response provided after such a change will be rejected since the knowledge no longer complies with the issued commitment. This might jeopardize the client's trust in the server: The client only perceives that

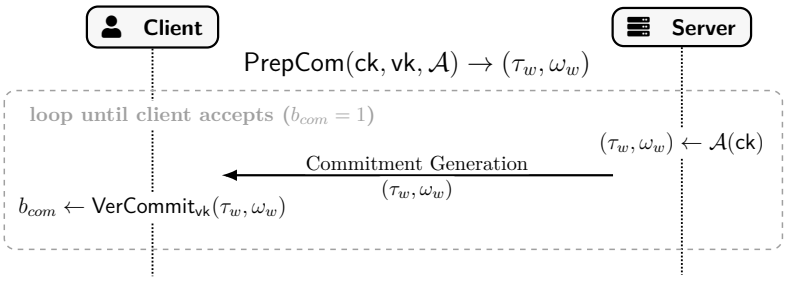

Figure 10: Commitment preprocessing.

the response is malformed but not the cause thereof. Hence, it might mistakenly assume the server to cheat and thereupon dismiss it as untrustworthy. This can be prevented by announcing changes in the knowledge publicly and by asking the client to restart the vPIR protocol in order to share a new commitment.

Let $\mathcal{A}$ be an efficient, server-controlled algorithm emulating the commitment algorithm Commit and let $(\mathsf{ck}, \mathsf{vk})$ be the key pair generated by $\mathsf{Com.Gen}$. Then, the commitment preprocessing phase $\mathsf{PrepCom}$ is defined by the data exchange depicted in Figure 10. Observe that the subscript w of the commitment τ_w and the opening information ω_w refers to the internal knowledge honest servers would commit to. Since, however, the server participating in the $\mathsf{PrepCom}$ protocol may be malicious, it can choose these data freely, i.e., independent of any actual knowledge. This is why the server in Figure 10 possesses no local input w.
Using $\mathsf{PrepCom}$, Figure 11 now defines the full interactive protocol between an honest party (client) and a malicious computing party (server) when utilizing a strongly vFHE scheme. In light of relaxing our security assumptions to honest commitments later on, we already include the case of weakly vFHE in the diagram. Note that the only changes occur during the commitment stage and are displayed side by side.

Remark 6.4. In the protocol of Figure 11, instead of letting the client set up the public and private parameters, one could also delegate the execution of Gen to a trusted third party. In that case, $(\mathsf{pk}, \mathsf{crs}, \mathsf{ck})$ would need to be distributed to both the client and the server, and $(\mathsf{sk}, \tau, \mathsf{vk})$ only to the client.

We now discuss the conditions on strongly vFHE schemes. They are similar to the requirements specified in Definitions 4.3 to 4.8 for general vFHEs. Indeed, the requirements on correctness, compactness and security directly transfer to this special case of vFHE and the precise statements follow from replacing the former function signatures with the corresponding new definitions. Nonetheless, out of these three, we again present the correctness condition because we use its precise formulation in a subsequent proof and want to emphasize that the new verification components do not interfere with this guarantee.

Definition 6.5 (Correctness for strongly vFHE)**.** Let λ be the security parameter, f be any function and $\mathcal{R}_f$ the corresponding relation of Definition 6.1. Let x be any valid input from the verifying party (trusted). Then, a strongly vFHE is *correct* if

$$\Pr\left[\mathsf{Dec}_{\mathsf{sk}}(c_y) \neq f(x) \;\middle|\; \begin{array}{r}((\mathsf{pk}, \mathsf{crs}, \mathsf{ck}), (\mathsf{sk}, \tau, \mathsf{vk})) \leftarrow \mathsf{Gen}(1^\lambda, \mathcal{R}_f) \\ c_x \leftarrow \mathsf{Enc}_{\mathsf{pk}}(x) \\ (\_, c_y, \_) \leftarrow \mathsf{Eval}_{\mathsf{pk}}(\mathsf{crs}, f, c_x, \_)\end{array}\right] \leq \mathsf{negl}(\lambda).$$

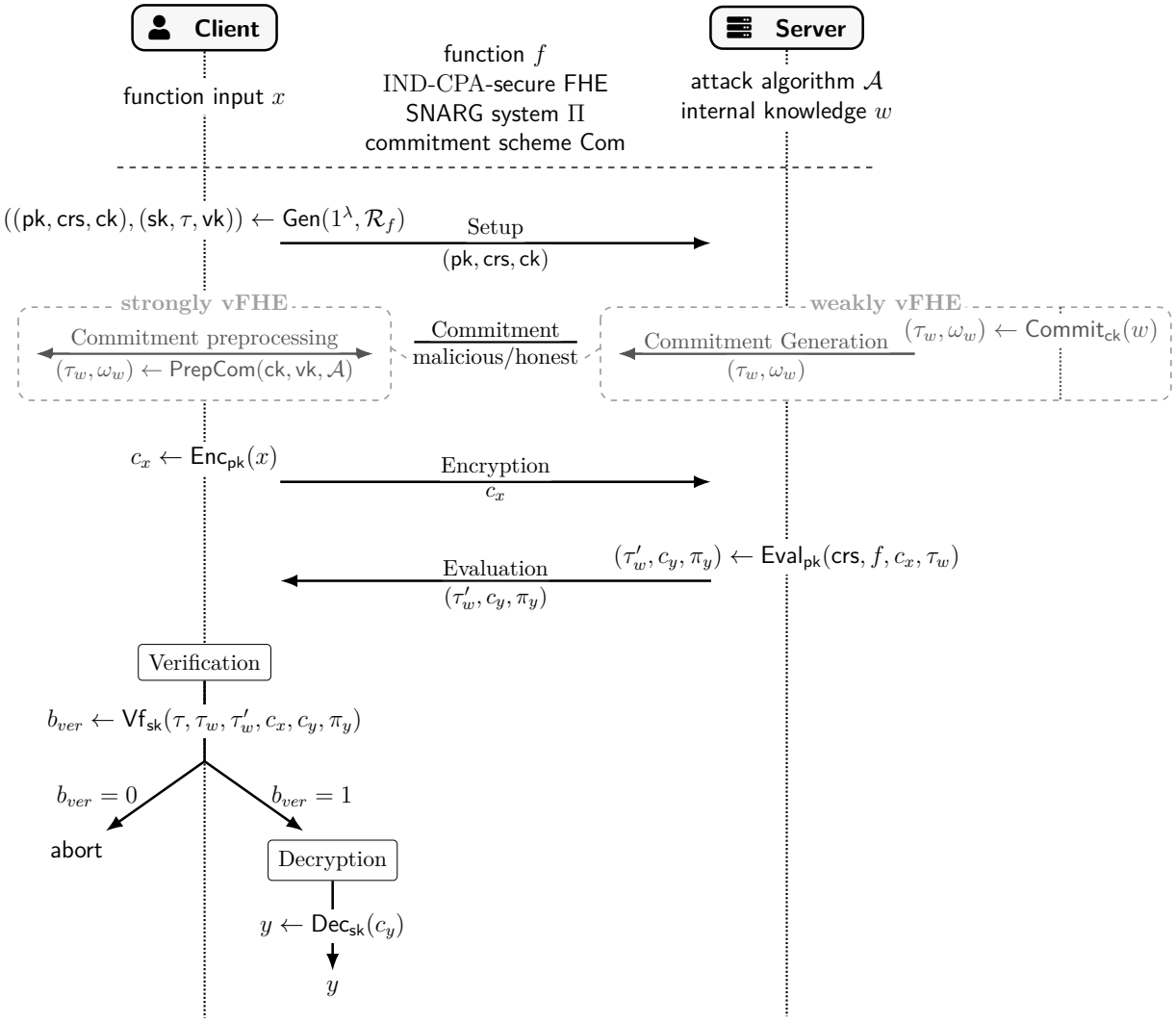

Figure 11: Interactive protocol for strongly and weakly vFHE, respectively.

We remark that, since correctness assumes the computing party to behave honestly, all commitment and verification steps are omitted.

Soundness requires slightly more adjustment. First, for strongly vFHE, the client and the computing party contribute inputs and knowledge to the evaluated function, respectively. In contrast, in the previous formulation of soundness, only the computing party provides inputs. Second, we no longer require the adversary to encrypt its contribution honestly. Instead, the interactive protocol $\mathsf{PrepCom}$ can handle manipulated knowledge commitments. Thus, we suggest the following reformulation of the soundness guarantee. It is inspired by the definition of knowledge soundness stated in [69] but relaxes it to the notion of adaptive soundness for argument systems [8], as opposed to knowledge systems.

Definition 6.6 (Soundness for strongly vFHE)**.** Let $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be any PPT adversary. Further, let f be any function taking inputs from two parties (one possibly modeled as internal knowledge) and let x be the input of the verifying party (trusted). If $\mathcal{R}_f$ is the relation of Definition 6.1 and $\mathcal{L}_f$ its associated language, *soundness for strongly vFHE* asks for

$$\Pr\left[\begin{array}{c}\mathsf{Vf}_{\mathsf{sk}}(\tau, \tau_w, \tau'_w, c_x, c_y, \pi_y) = 1 \\ \wedge\, (f, c_x, c_y, \tau_w) \notin \mathcal{L}_f\end{array} \;\middle|\; \begin{array}{r}((\mathsf{pk}, \mathsf{crs}, \mathsf{ck}), (\mathsf{sk}, \tau, \mathsf{vk})) \leftarrow \mathsf{Gen}(1^\lambda, \mathsf{R}_f) \\ (\tau_w, \_) \leftarrow \mathsf{PrepCom}(\mathsf{ck}, \mathsf{vk}, \mathcal{A}_1^{\mathcal{O}_{\mathsf{Vf}}}) \\ c_x \leftarrow \mathsf{Enc}_{\mathsf{pk}}(x) \\ (\tau'_w, c_y, \pi_y) \leftarrow \mathcal{A}_2^{\mathcal{O}_{\mathsf{Vf}}}(\mathsf{pk}, \mathsf{crs}, \mathsf{ck}, \tau_w, c_x)\end{array}\right] \leq \mathsf{negl}(\lambda).$$

Remark 6.7. We elaborate on the comparison between this new soundness condition for strongly vFHE and the general one of Definition 4.7. For clearer reasoning, we refer to the attacker of the general soundness requirement as $\mathcal{A}$ and to the adversary in Definition 6.6 as $\mathcal{B}$.

- **Strength of guarantee** Comparing the two definitions in terms of strength is not immediate. To approach a conclusion, we first transfer the attacker $\mathcal{A}$ to the setting of strongly vFHE. Hereby, we identify three main capabilities of $\mathcal{A}$. First, it has control over the entire function input. Second, it is bound to provide honest encryptions and verification tags of its input. Third, the final response and all verification data associated with it can be chosen arbitrarily. This translates to: $\mathcal{A}$ controls both the client input x as well as the internal knowledge w. It is forced to encrypt x honestly, yielding c_x, and to publish an honest commitment to w. Moreover, it may guess the response c_y, the correctness proof π_y and the updated commitment τ'_w freely. This does not quite match the scope of the new attacker $\mathcal{B}$. The main difference is that $\mathcal{B}$ can issue tampered commitments and the $\mathsf{PrepCom}$ procedure ensures that, up to some negligible probability, the proper vPIR protocol is only executed once the client is convinced of the correctness of the commitment. $\mathcal{A}$ on the other hand, may only submit commitments that are guaranteed to be correct. All other steps of soundness for strongly vFHE, however, can be emulated by the attacker $\mathcal{A}$. To imitate the client input, for example, $\mathcal{A}$ can encrypt its input x honestly and ignore it whenever choosing parameter values maliciously. $\mathcal{B}$'s random decision on the response ciphertext c_y and all related verification data directly carries over to the attacker $\mathcal{A}$. In summary, once we assume honest commitments, as will be the case for weakly vFHE, we can view the new soundness requirement as a weaker

version of the previous definition. In particular, all vFHE protocols meeting the stricter requirement on general vFHE schemes are inherently weakly vFHE schemes and can therefore be used for our construction. However, as long as we keep to strongly vFHE, we have to be slightly careful and check whether a vFHE scheme in the general sense also achieves soundness when faced with malformed commitments.

- **Conditioned event** Another novelty as compared to Definition 4.7, is the check whether or not the statement (f, c_x, c_y, τ_w) is part of the language $\mathcal{L}_f$. If not, there is no witness w that can attest that, given the commitment τ_w, c_y is the correct result of homomorphically evaluating f on c_x. Transferring this to the level of plaintexts, this means that decrypting c_y cannot result in the desired value $f(x)$. This is precisely what is checked in the previous soundness definition. Vice versa, if $\mathsf{Dec}_{\mathsf{sk}}(c_y)$ is not $f(x)$, then by correctness of the decryption and the evaluation function of the underlying FHE scheme, c_y is not the evaluation of f on c_x, regardless of the witness. Consequently, (f, c_x, c_y, τ_w) cannot belong to the language $\mathcal{L}_f$.

As we also discuss vPIR constructions from weakly vFHE later on, we briefly mention the adaptations required to adjust the above definition to the honest digest setting. To define soundness in this context, merely replace

$$(\tau_w, \_) \leftarrow \mathsf{PrepCom}(\mathsf{ck}, \mathsf{vk}, \mathcal{A}_1^{\mathcal{O}_{\mathsf{Vf}}})$$

by the two lines

$$\begin{aligned} w &\leftarrow \mathcal{A}_1^{\mathcal{O}_{\mathsf{Vf}}}() \\ (\tau_w, \_) &\leftarrow \mathsf{Commit}_{\mathsf{ck}}(w) \end{aligned}$$

and use τ_w in place of τ'_w.
Obviously, this constitutes an even weaker notion of soundness as it forces adversaries to commit to their knowledge honestly by using the designated Commit algorithm.

6.1.2 Protocol for vFHE-based vPIR

Now, everything is in place to define verifiable PIR schemes. To avoid redundancy, we first concentrate on the setting that tolerates maliciously generated commitments. This setting requires slightly more care and provides stronger and therefore more desirable guarantees than the analogue for honest commitments. Indeed, it is a simple consequence to arrive at the definition of vPIR scheme with honest digests.

Definition 6.8 (Verifiable PIR from strongly vFHE fit for SNARGs)**.** Let $DB = (DB_1, \ldots, DB_n)$ be a database of n files, each of size m. Interpret DB as a matrix

$$DB = \{DB_{j,s}\}_{\substack{j=1,\ldots,n \\ s=1,\ldots,m}} \in \{0,1\}^{n \times m}.$$

Let f be the functionality realizing the vector-PIR protocol. Further, let $(\mathsf{Gen}, \mathsf{Commit}, \mathsf{VerCommit}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{Vf}, \mathsf{Dec})$ be a strongly vFHE scheme that supports vector operations (addition, multiplication and rotation). Taking inspiration from [50, 60], define the

relation

$$\mathcal{R}_f \coloneqq \{((f, \{\widehat{a}_k\}_k, r, d), (d', \{DB_{\cdot,s}\}_s, \{\widehat{b}_k\}_k)) \mid r = \mathsf{Eval}_{\mathsf{pk}}\left(f, \{\widehat{a}_k\}_k, \{\widehat{b}_k\}_k, \{\widehat{DB}_{\cdot,s}\}_s\right) \\ \wedge\ \widehat{DB}_{\cdot,s} = \mathsf{Enc}_{\mathsf{pk}}(DB_{\cdot,s})\, \forall s \in [m] \\ \wedge\ (d', \_) = \mathsf{Digest}(\mathsf{pp}, \{DB_{\cdot,s}\}_s) \\ \wedge\ d = d'\}.$$

Based on Definition 6.1, we define the vPIR constructed from strongly vFHE as

- $\mathsf{Setup}(1^\lambda, \mathcal{R}_f) \to (\mathsf{pp}, \mathsf{sp})$: The client runs the strongly vFHE generation algorithm $((\mathsf{pk}, \mathsf{crs}, \mathsf{ck}), (\mathsf{sk}, \tau, \mathsf{vk})) \leftarrow \mathsf{Gen}(1^\lambda, \mathcal{R}_f)$. Set $\mathsf{pp} \coloneqq (\mathsf{pk}, \mathsf{crs}, \mathsf{ck})$ and $\mathsf{sp} \coloneqq (\mathsf{sk}, \tau, \mathsf{vk})$. Publish pp and keep sp private.

- $\mathsf{Digest}(\mathsf{pp}, DB) \to (d, \omega)$: Parse $\mathsf{pp} = (\mathsf{pk}, \mathsf{crs}, \mathsf{ck})$. As preprocessing, the server computes the binary representations $(\beta_{j,1}, \ldots, \beta_{j,\ell})$ of all indices $j \in [n]$ and sets the vectors $b_k \coloneqq (\beta_{1,k}, \ldots, \beta_{n,k}), \forall k \in [\ell]$. Let $\widehat{b}_k \coloneqq \mathsf{Enc}_{\mathsf{pk}}(b_k)$. Next, it produces the database digest $(d, \omega) \leftarrow \mathsf{Commit}_{\mathsf{ck}}(\{DB_{\cdot,s}\}_s, \left\{\widehat{b}_k\right\}_k)$, where the database columns $\{DB_{\cdot,s}\}_s$ together with the encrypted index set vectors $\left\{\widehat{b}_k\right\}_k$ are the internal knowledge. Send (d, ω) to the client.

- $\mathsf{VerDigest}(d, \omega) \to b_{dig}$: To verify the received digest d, execute the algorithm $b_{dig} \coloneqq \mathsf{VerCommit}_{\mathsf{vk}}(d, \omega)$. Output the resulting decision bit b_{dig}.

- $\mathsf{Query}(\mathsf{pp}, d, i) \to q$: Parse $\mathsf{pp} = (\mathsf{pk}, \mathsf{crs}, \mathsf{ck})$. The client expands its query index $i \in [n]$ in binary, $i = (\alpha_1, \ldots, \alpha_\ell)$, and encrypts the vectors $a_k \coloneqq (\alpha_k, \ldots, \alpha_k) \in \{0,1\}^n$, obtaining $\widehat{a}_k \coloneqq \mathsf{Enc}_{\mathsf{pk}}(a_k)$. Send $q \coloneqq (d, \{\widehat{a}_k\}_{k=1}^\ell)$ as query to the server.

- $\mathsf{Answer}(\mathsf{pp}, DB, q) \to a$: To compute the response, the server first encrypts the database column-wise, i.e., $\widehat{DB}_{\cdot,s} \coloneqq \mathsf{Enc}_{\mathsf{pk}}(DB_{\cdot,s}), \forall s \in [m]$. These encryptions are then input to homomorphically evaluate the functionality f, yielding $(d', r, \pi_r) \coloneqq \mathsf{Eval}_{\mathsf{pk}}(\mathsf{crs}, f, q)$. Hereby, d' is the newly computed digest, r the evaluation result and π_r the correctness proof. Output $a \coloneqq (d', r, \pi_r)$.

- $\mathsf{Decryption}(q, a) \to out$: Parse $a = (d', r, \pi_r)$ and $q = (d, \{\widehat{a}_k\}_{k=1}^\ell)$. Verify the answer $b_{ver} \coloneqq \mathsf{Vf}_{\mathsf{sk}}(\tau, d, d', \{\widehat{a}_k\}_{k=1}^\ell, r, \pi_r)$. If verification fails ($b_{ver} = 0$), abort. If it succeeds, decrypt and output $out \coloneqq \mathsf{Dec}_{\mathsf{sk}}(r)$.

Remark 6.9. For honest servers, it is more efficient to perform the database encryption as part of a preprocessing phase. Only when the database is updated its encryption has to be changed accordingly. Otherwise, it can be reused for all subsequent client queries. Similarly, generating the encryptions $\left\{\widehat{b}_k\right\}_k$ of the index set vectors could also be moved from the digest algorithm to a preprocessing stage, thereby reducing computation costs in the online phase.

Similarly to the case of strongly vFHE, one can define an interactive protocol for a meaningful vPIR protocol. Reminiscent of $\mathsf{PrepCom}$, it also makes use of a preprocessing stage, called $\mathsf{PrepDigest}$. Both the preprocessing procedure and the interactive protocol for vPIR

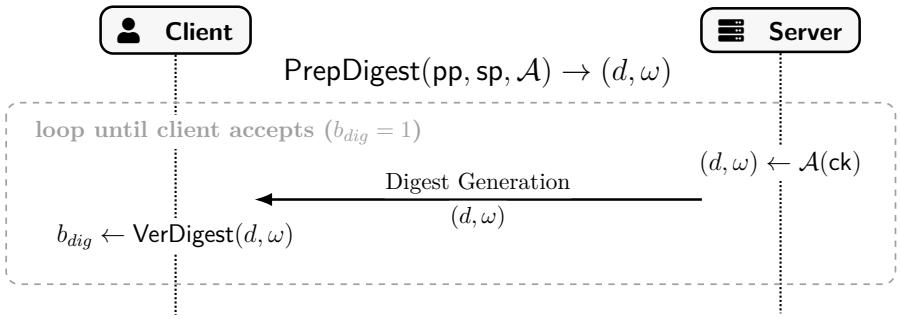

Figure 12: Digest preprocessing.

arise from the analogues for strongly vFHE (see the procedures in Figures 10 and 11) in the obvious way – replace each algorithm with its vPIR counterpart. We visualize the digest preprocessing in Figure 12 and the complete vPIR-from-vFHE protocol in Figure 13. Observe that the vPIR-from-strongly vFHE protocol is mostly identical to the general vPIR protocol depicted in Figure 5. The additional computation steps stemming from the SNARG verification are encapsulated within the standard algorithms and therefore hidden in the structural outline of the scheme.

6.2 Requirement Transfer

So far, we proposed a construction of verifiable PIR with verifiable FHE as the underlying component, to meet the expectations on security and to verify the correctness of the server-generated response. It remains to show that the scheme described exhibits all desirable properties of vPIR schemes, formally stated in Definitions 3.3 to 3.6. For a better understanding, Table 5 provides an overview of all parameters and algorithms of strongly vFHE schemes and their counterparts for the vPIR scheme arising from it. We call this vPIR scheme *vPIR-from-strongly vFHE*. Note that for brevity, we omit writing the ranges of the variables repetitively. This is, in the table, we always have $j \in [n]$, $s \in [m]$ and $k \in [\ell]$ for $\ell = \lceil \log(n) \rceil$.

Table 5: Parameters and algorithms of strongly vFHE and vPIR-from-strongly vFHE side by side.

Parameter/Algorithm	Strongly vFHE	vPIR-from-strongly vFHE
evaluated function	f (any)	f (vPIR functionality, see Procedure 1)
public parameters	$(\mathsf{pk}, \mathsf{crs}, \mathsf{ck})$ pk: public encryption key crs: common reference string ck: commitment key	pp
secret parameters	$(\mathsf{sk}, \tau, \mathsf{vk})$ sk: secret decryption key τ: verification tag vk: verification key	sp

continued on the next page

Table 5 – *continued from the previous page*

Parameter/Algorithm	Strongly vFHE	vPIR-from-strongly vFHE
server's knowledge	w	DB; $\mathcal{I}(DB) = \{j\}_{j=1}^{n}$ column representation $\{DB_{\cdot,s}\}_s$ binary representations $\{j\}_j = \{(\beta_{j,1}, \ldots, \beta_{j,\ell})\}_j$ index set vectors $\{b_k\}_k = \{(\beta_{1,k}, \ldots, \beta_{n,k})\}_k$ encrypted vectors $\left\{\widehat{b}_k\right\}_k = \{\mathsf{Enc}_{\mathsf{pk}}(b_k)\}_k$
commitment/digest	τ_w	d
(updated)	τ'_w	d'
opening	ω_w	ω
verify-commitment bit	b_{com}	b_{dig}
client input	x	$i \in [n]$ binary representation $i = (\alpha_1, \ldots, \alpha_\ell)$ plaintext vectors $\{a_k\}_k = \{(\alpha_k, \ldots, \alpha_k)\}_k$ query vector $\{\widehat{a}_k\}_k = \{\mathsf{Enc}_{\mathsf{pk}}(a_k)\}_k$
challenge/query	c_x; (τ_w, c_x)	$\{\widehat{a}_k\}_k$; $q = (d, \{\widehat{a}_k\}_k)$
evaluation outcome	c_y	r
correctness proof	π_y	π_r
server's answer	(τ'_w, c_y, π_y)	$a = (d', r, \pi_r)$
verification bit	b_{ver}	b_{ver}
decryption outcome	y	out
.....	.....	.....
setup	$((\mathsf{pk}, \mathsf{crs}, \mathsf{ck}), (\mathsf{sk}, \tau, \mathsf{vk}))$ $\leftarrow \mathsf{Gen}(1^\lambda, \mathcal{R}_f)$	$(\mathsf{pp}, \mathsf{sp}) \leftarrow \mathsf{Setup}(1^\lambda, \mathcal{R}_f)$
commitment/digest	$(\tau_w, \omega_w) \leftarrow \mathsf{Commit}_{\mathsf{ck}}(w)$	$(d, \omega) \leftarrow \mathsf{Digest}(\mathsf{pp}, DB)$
verify commitment	$b_{com} \leftarrow \mathsf{VerCommit}_{\mathsf{vk}}(\tau_w, \omega_w)$	$b_{dig} \leftarrow \mathsf{VerDigest}(d, \omega)$
encryption/query	$c_x \leftarrow \mathsf{Enc}_{\mathsf{pk}}(x)$	$q \leftarrow \mathsf{Query}(\mathsf{pp}, d, i)$
evaluation/answer	$(\tau'_w, c_y, \pi_y) \leftarrow \mathsf{Eval}_{\mathsf{pk}}(\mathsf{crs}, f, c_x, \tau_w)$	$a \leftarrow \mathsf{Answer}(\mathsf{pp}, DB, q)$
verification and/or	$b_{ver} \leftarrow \mathsf{Vf}_{\mathsf{sk}}(\tau, \tau_w, \tau'_w, c_x, c_y, \pi_y)$	$out \leftarrow \mathsf{Decryption}(q, a)$
decryption	$y \leftarrow \mathsf{Dec}_{\mathsf{sk}}(c_y)$	

6.2.1 Correctness

First, we show correctness of the constructed vPIR-from-strongly vFHE scheme. We remark that the subsequent proof ignores the digest generation as well as the verification procedure in the correctness definition of general vPIR schemes (see Definition 3.3) since the server is assumed to comply with the protocol. That omitting these checks does not weaken the expressiveness of the correctness statement has already been pointed out when adapting the correctness definition of strongly vFHE (see Definition 6.5).

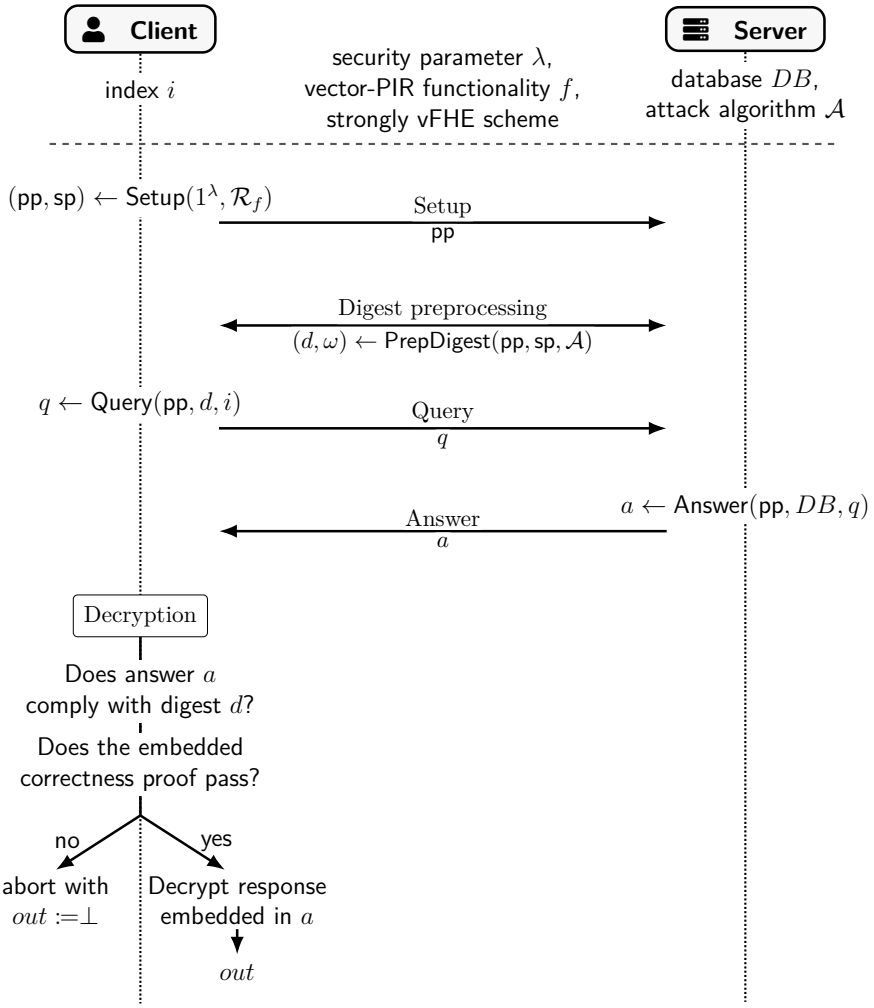

Figure 13: Interactive protocol for vPIR constructed from strongly vFHE.

Theorem 6.10. *The vPIR-from-strongly vFHE scheme described in Definition 6.8 is correct. Hereby, we refer to Definition 6.5 as the appropriate notion of correctness for the underlying strongly vFHE scheme.*

Proof. We proceed by contradiction. Assume the vPIR-from-strongly vFHE scheme is not correct in the sense of Definition 3.3 (without the digest generation and verification step). Then there exists an index $i \in [n]$ such that running the vPIR protocol honestly yields an output $out \neq DB_i$, where DB_i is the i^{th} row of DB. With the help of Table 5, we unfold the precise definitions of the vPIR parameters in the context of our construction.

- The public parameters consist of the encryption key, the common reference string and the commitment key. That is, $\mathsf{pp} = (\mathsf{pk}, \mathsf{crs}, \mathsf{ck})$.
- As query, the client sends the encrypted vectors $q = (\_, \{\widehat{a}_k\}_k)$ computed from its index of interest i.
- The server responds with $a = (\_, r, \pi_r)$, where
  - $r \coloneqq \mathsf{Eval}_{\mathsf{pk}}\left(f, \{\widehat{a}_k\}_k, \left\{\widehat{b}_k\right\}_k, \left\{\widehat{DB}_{\cdot,s}\right\}_s\right)$ is the homomorphic evaluation of the PIR functionality f over the client's query $\{\widehat{a}_k\}_{k=1,\ldots,\ell}$ and the server's internal knowledge $\left\{\widehat{b}_k\right\}_{k=1,\ldots,\ell}$ and $\left\{\widehat{DB}_{\cdot,s}\right\}_{s=1,\ldots,m}$.
  - π_r is the SNARG proof for $\left((f, \{\widehat{a}_k\}_k, r, \_), (\_, \{DB_{\cdot,s}\}_s, \left\{\widehat{b}_k\right\}_k)\right) \in \mathcal{R}_f$, where $\mathcal{R}_f$ is the relation of Definition 6.8.
- Since the correctness requirement assumes honest computation, we know that the answer a passes verification, that is $\mathsf{Vf}_{\mathsf{sk}}(\tau, q, a) = 1$, and hence, the decryption algorithm proceeds with decrypting the evaluation result r, yielding $out \coloneqq \mathsf{Dec}_{\mathsf{sk}}(r)$.

By assumption, $\mathsf{Dec}_{\mathsf{sk}}(r) = out \neq DB_i \overset{(*)}{=} f(\{a_k\}_k, \{b_k\}_k, \{DB_{\cdot,s}\}_s)$. Hereby, $(*)$ holds by the definition of the PIR functionality f. However, this contradicts the correctness guarantee of vFHE, ensuring that $\mathsf{Dec}_{\mathsf{sk}}(r) = f(x)$, for $x \coloneqq \left\{\{a_k\}_k, \{b_k\}_k, \{DB_{\cdot,s}\}_s\right\}$. □

6.2.2 Integrity

We continue the requirement transfer by proving that the constructed vPIR scheme inherits its integrity guarantee from the soundness requirement on the underlying strongly vFHE scheme. To this end, we first present the integrity definition as it reads after adjusting it to the notation for vPIR-from-strongly vFHE. Note that its expressiveness remains the same as in Definition 3.4 and only the concrete in- and output parameters change.

Definition 6.11 (Integrity for vPIR-from-strongly vFHE)**.** Like for the general definition of integrity in Definition 3.4, let $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be some efficient PPT adversary emulating the digest and answer generation. Let the setup $(\mathsf{pp} = (\mathsf{pk}, \mathsf{crs}, \mathsf{ck}), \mathsf{sp} = (\mathsf{sk}, \tau, \mathsf{vk})) \leftarrow \mathsf{Setup}(1^\lambda, \mathcal{R}_f)$ be performed by a trusted party. Then, for every digest output by the digest preprocessing $(d, \omega) \leftarrow \mathsf{PrepDigest}(\mathsf{pp}, \mathsf{sp}, \mathcal{A}_1)$, we require that

$$\Pr\left[\begin{array}{c|c} \exists \text{ a database } DB & q = (d, \{\widehat{a}_k\}_k) \leftarrow \mathsf{Query}(\mathsf{pp}, d, i) \\ \text{such that } out \notin \{DB_i, \bot\}, & a = (d', r, \pi_r) \leftarrow \mathcal{A}_2(\mathsf{pp}, q) \\ \text{for any index } i. & out \in \{\mathsf{Dec}_{\mathsf{sk}}(r), \bot\} \leftarrow \mathsf{Decryption}(d, a) \end{array}\right] \leq \mathsf{negl}(\lambda).$$

We now show, that this integrity formulation is implied by soundness for strongly vFHE.

Theorem 6.12. *The vPIR-from-strongly vFHE scheme specified in Definition 6.8 fulfills the integrity requirement of Definition 6.11.*

Proof. Again, we proceed by contradiction. Assume that there is an attacker $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ for the integrity requirement of vPIR and that the probability stated in Definition 6.11 is non-negligible. We sketch how to use $\mathcal{A}$ to define an attacker $\mathcal{B} = (\mathcal{B}_1, \mathcal{B}_2)$ violating the soundness condition of strongly verifiable FHE, see Definition 6.6.
In fact, we reason that it suffices to set

$$\mathcal{B}_1^{\mathcal{O}_{\mathsf{Vf}}}() \coloneqq \mathcal{A}_1(),$$
$$\mathcal{B}_2^{\mathcal{O}_{\mathsf{Vf}}}(\tau_w, c_x) \coloneqq \mathcal{A}_2(q).$$

To see why this definition is reasonable, recall that the query is of the form $q = (d, \{\widehat{a}_k\}_k)$, where $(d, \_) \coloneqq \mathsf{PrepDigest}(\mathsf{pp}, \mathsf{sp}, \mathcal{A}_1)$. Hence, when considering $x \coloneqq \{a_k\}_k$ and identifying $\mathsf{Enc}_{\mathsf{pk}}(\{a_k\}_k) = \{\mathsf{Enc}_{\mathsf{pk}}(a_k)\}_k$ (see Subsection 2.1), the tuple (τ_w, c_x) with ciphertext $c_x \coloneqq \mathsf{Enc}_{\mathsf{pk}}(x)$ and commitment $\tau_w \coloneqq \mathcal{B}_1^{\mathcal{O}_{\mathsf{Vf}}}()$ is equivalent to q.
With this attacker $\mathcal{B}$ at hand, first observe that, under our definition of vPIR-from-strongly vFHE, the integrity and soundness requirements come with identical contexts for the events over which the probabilities are calculated. Both start with generating keys for the encryption scheme and the verification method: Let the client set up the strongly vFHE computation by

$$((\mathsf{pk}, \mathsf{crs}, \mathsf{ck}), (\mathsf{sk}, \tau, \mathsf{vk})) \leftarrow \mathsf{Gen}(1^\lambda, \mathcal{R}_f).$$

Tracing parameters, this is equivalent to computing $(\mathsf{pp}, \mathsf{sp}) \leftarrow \mathsf{Setup}(1^\lambda, \mathcal{R}_f)$. By writing out the definitions for the other vPIR parameters in a similar way, it becomes clear that the conditional contexts agree via our construction.
With this in mind, we argue that a violation of integrity implies a violation of soundness: Let i be the queried index and let DB be the database such that $out \notin \{DB_i, \perp\}$. By assumption, this database exists with non-negligible probability. Further, let $(d, \_) \leftarrow \mathsf{PrepDigest}(\mathsf{pp}, \mathsf{sp}, \mathcal{A}_1)$ be the database digest issued initially. Since the preprocessing stage is able to deal with malicious digests, we can assume that $(d, \_) = \mathsf{Digest}(\mathsf{pp}, \{DB_{\cdot,s}\}_s)$ is the honest digest of DB. Further, assume r to be the server's response leading to the outcome out, i.e., $out = \mathsf{Dec}_{\mathsf{sk}}(r)$. Then, the data $\{i, d, out\}$ corresponds to the statement

$$X \coloneqq (f, \{\widehat{a}_k\}_k, r, d) = (f, c_x, c_y, \tau_w).$$

We reason that $\mathsf{Vf}_{\mathsf{sk}}(\tau, q, a) = \mathsf{Vf}_{\mathsf{sk}}(\tau, \tau_w, \tau'_w, c_x, c_y, \pi_y) = 1$ and $X \notin \mathcal{L}_f$, which concludes the proof because DB exists with non-negligible probability.
Since $out \neq \perp$, verification must have been successful. Hence, $\mathsf{Vf}_{\mathsf{sk}}(\tau, q, a) = 1$ as desired.
For the second part of the claim, assume for contradiction that $X \in \mathcal{L}_f$. This means that there is a witness tuple

$$W \coloneqq (\widetilde{\tau}_w, w) = \left(\widetilde{d}, \left\{\widetilde{DB}_{\cdot,s}\right\}_s, \left\{\widehat{\widetilde{b}}_k\right\}_k\right)$$

such that $(X, W) \in \mathcal{R}_f$. Belonging to the relation means that

$$\begin{aligned} r &= \mathsf{Eval}_{\mathsf{pk}}\left(f, \{\widehat{a}_k\}_k, \left\{\widehat{\widetilde{b}}_k\right\}_k, \left\{\widehat{\widetilde{DB}}_{\cdot,s}\right\}_s\right), \\ \widehat{\widetilde{DB}}_{\cdot,s} &= \mathsf{Enc}_{\mathsf{pk}}\left(\widetilde{DB}_{\cdot,s}\right), \forall s \in [m], \\ (\widetilde{d}, \_) &= \mathsf{Digest}\left(\mathsf{pp}, \left\{\widetilde{DB}_{\cdot,s}\right\}_s\right), \\ d &= \widetilde{d}. \end{aligned}$$

Recall that by definition of the PIR functionality f, correct evaluation simplifies the first equation to $r = \mathsf{Enc}_{\mathsf{pk}}(\widetilde{DB}_i)$. By correctness of the underlying FHE scheme, this yields

$$out = \mathsf{Dec}_{\mathsf{sk}}(r) = \mathsf{Dec}_{\mathsf{sk}}\left(\mathsf{Enc}_{\mathsf{pk}}(\widetilde{DB}_i)\right) = \widetilde{DB}_i.$$

Therefore, $DB_i \neq out = \widetilde{DB}_i$, which shows that DB and $\widetilde{DB}$ are distinct databases. However, we also have that

$$\mathsf{Digest}\left(\mathsf{pp}, \left\{\widetilde{DB}_{\cdot,s}\right\}_s\right) = (\widetilde{d}, \_) = (d, \_) = \mathsf{Digest}\left(\mathsf{pp}, \{DB_{\cdot,s}\}_s\right).$$

This directly contradicts the binding property of commitment schemes, which guarantees that no two distinct databases have the same commitment. Hence, $X \notin \mathcal{L}_f$ as claimed. □

6.2.3 Privacy

To arrive at a vPIR scheme that deserves this name, it is left to prove that vPIR-from-strongly vFHE also meets the requirement on malicious privacy.

Theorem 6.13. *The vPIR-from-strongly vFHE scheme specified in Definition 6.8 fulfills the malicious privacy requirement of Definition 3.6.*

Proof. Proof by contradiction: Assume malicious privacy does not hold. Hence, there exists an adversary $\mathcal{A}$, a database DB of length n and a query index $i \in [n]$ such that for every efficient simulating PPT algorithm Sim, the distributions given in Definition 3.6 are distinguishable. We introduce the superscripts Sim whenever referring to parameters that are part of the simulating distribution.
Define a simulator Sim as

- $\mathsf{Sim.Setup}(1^\lambda, \mathcal{R}_f) = \mathsf{Setup}(1^\lambda, \mathcal{R}_f)$
- $\mathsf{Sim.Query}(\mathsf{pp}, d) =$
  - Randomly choose $j \xleftarrow{\$} [n]$.
  - Run $\mathsf{Query}(\mathsf{pp}, d, j)$.
- $\mathsf{Sim.Decryption}(q, a) = \mathsf{Decryption}(q, a)$.

Tracing the data input to the adversary of Definition 3.6, one sees that the decision bit b of algorithm $\mathcal{A}_3$ is based on pp, DB, d, q and $\mathbb{1}\{out = \perp\}$. Similarly for $\mathcal{A}_3^{\mathsf{Sim}}$. Since $\mathsf{pp} = \mathsf{pp}^{\mathsf{Sim}}, DB = DB^{\mathsf{Sim}}$ and $d = d^{\mathsf{Sim}}$ (these parameters are not influenced by the simulator), distinguishability of b and b^{Sim} must be caused by either q or $\mathbb{1}\{out = \perp\}$.

(i) Suppose, q and q^{Sim} cause the distinguishability. Recall that

- $q = (d, \{\widehat{a}_k\}_k)$ for $a_k = (\alpha_k, \ldots, \alpha_k)$, where $i = (\alpha_1, \ldots, \alpha_\ell)$ is the binary representation of i.
- $q^{\mathsf{Sim}} = \left(d^{\mathsf{Sim}}, \left\{\widehat{a'}_k\right\}_k\right)$ for $a'_k = (\alpha'_k, \ldots, \alpha'_k)$, where $j = (\alpha'_1, \ldots, \alpha'_\ell)$ is the binary representation of j, which is randomly chosen in $\mathsf{Sim.Query}$. Since distinguishability is assumed, without loss of generality $j \neq i$.

By a slight abuse of notation, write $q = \mathsf{Enc}_{\mathsf{pk}}(i) =: c_i$ and $q^{\mathsf{Sim}} = \mathsf{Enc}_{\mathsf{pk}}(j) =: c_j$.

Now, let $\mathcal{B}$ be the attacker in the IND-CCVA-security game of Figure 7. $\mathcal{B}_1$ may choose $m_0 = i$ and $m_1 = j$. Since q and q^{Sim} are assumed to be distinguishable, the advantage $\mathsf{Adv}^{\text{IND-CCVA}}_{\mathcal{B}}(\lambda)$ must be noticeable. This contradicts vFHE security.

(ii) Suppose that the questions on abort, i.e., $\mathbb{1}\{out = \perp\}$ and $\mathbb{1}\left\{out^{\mathsf{Sim}} = \perp\right\}$ cause the distinguishability. Again, the attack algorithm $\mathcal{B}_1$ in the IND-CCVA-security game can play it with messages $m_0 = i$ and $m_1 = j$. Let $\mathcal{B}_2$ perform the following operations on the challenge $c^* \in \{c_i, c_j\}$.

- Run the vPIR adversary $\mathcal{A}_2$ on c^*, obtaining an answer a^*.
- Decrypt $out^* \leftarrow \mathsf{Decryption}(q, a^*)$.
- Query the verification oracle with out^*.

Since we assumed the questions on abort to tell apart i and j, the vFHE attacker $\mathcal{B}$ can recover the plaintext associated to the challenge c^*, contradicting IND-CCVA security of the underlying strongly vFHE scheme.

In conclusion, we get contradictions in all cases. □

Combining Theorems 6.10,6.12 and 6.13, we have proven that, under some carefully chosen structural assumptions, verifiable FHE can serve as a starting point to construct verifiable PIR. This leads to our main result.

Main Theorem 1 (vPIR from strongly vFHE)**.** *The vPIR-from-strongly vFHE scheme as defined in Definition 6.8 is a verifiable PIR scheme in the sense of Definition 3.1, meeting the requirements on correctness, integrity and malicious privacy.*

6.2.4 Adaptations for Weakly vFHE

So far, we have shown that one can construct vPIR schemes that are capable of handling malicious digests from strongly vFHE schemes. Many of the existing vPIR protocols, however, are limited to tolerating honest digests; see Table 2 for examples. To arrive at this weaker notion of verifiable PIR, we now show that it suffices to base the construction on the weaker analogue of vFHE, namely weakly vFHE. Observe, that its precise definition can easily be derived from strongly vFHE by replacing the commitment preprocessing $\mathsf{PrepCom}$ with the honest commitment generation Commit. This is made apparent in the interactive protocol for both strongly and weakly vFHE in Figure 11. Then, the above construction carries through with only minor adaptations required. Indeed, most arguments are identical to before. Thus, we merely sketch the adapted proofs and skip them

altogether in case they only differ in notation. Before detailing the transfer of requirements, we fix one further terminology: We say *vPIR-from-weakly vFHE* when referring to vPIR schemes constructed from vFHE schemes that assume honest commitments.

Correctness Since the correctness definition does not make use of commitments or digests, Theorem 6.10 also holds in our limited setting by the same reasoning as before.

Corollary 6.14. *The vPIR-from-weakly vFHE schemes arising from our construction are correct.*

Integrity Observe that at the end of the digest preprocessing phase $\mathsf{PrepDigest}$, the client holds a digest that successfully passes the digest verification. Therefore, up to some negligible probability, the resulting digest can be assumed to be correct. Hence, by replacing the preprocessing protocol with the honest digest generation algorithm, we reach the setting of vPIR-from-weakly vFHE. Moreover, all previous arguments establishing the integrity of vPIR-from-strongly vFHE have already assumed that the digest produced by $\mathsf{PrepDigest}$ is correct. This shows that, by merely dropping the first attacker algorithms $\mathcal{A}_1$ and $\mathcal{B}_1$ in the proof of Theorem 6.12 and by using Digest whenever we previously had $\mathsf{PrepDigest}$, we obtain a proof of the following corollary.

Corollary 6.15. *The vPIR-from-weakly vFHE schemes arising from our construction fulfill the requirement on integrity for honest digests.*

Privacy To adapt malicious privacy (see Definition 3.6) to vPIR schemes with honest digests, merely exchange $d \leftarrow \mathcal{A}_1(\mathsf{pp}, DB)$ with $(d, \omega) \leftarrow \mathsf{Digest}(\mathsf{pp}, DB)$. Crucially, this leaves the simulator algorithms and their counterparts for the real computation unchanged. The definition of IND-CCVA security remains the same for weakly instead of strongly vFHE. Therefore, proving that IND-CCVA security implies malicious privacy for honest digests is analogous to before.

Corollary 6.16. *The vPIR-from-weakly vFHE schemes arising from our construction fulfill the requirement on malicious privacy for honest digests.*

Overall, from Corollaries 6.14, 6.15 and 6.16, we can conclude, like before, that all properties of the underlying weakly vFHE scheme are preserved by the construction.

Main Theorem 2 (vPIR from weakly vFHE)**.** *The vPIR-from-weakly vFHE schemes arising from our construction are verifiable PIR schemes in the sense of Definition 3.1 after adapting it to the use of honestly generated digests. Moreover, they meet the requirements on correctness, integrity and malicious privacy for honest digests.*

7 Conclusion and Outlook

Interacting with parties online comes with certain risks. Since service providers are often distant entities, their true incentives are usually unknown and might conflict with the clients' interests. In the context of data retrieval and outsourced computation, cryptographic primitives like Private Information Retrieval (PIR) and Fully Homomorphic Encryption (FHE) aim at protecting clients in the presence of adversaries. Ideally, these protections withstand any kind of fraudulent behavior, including active attempts of security breaches and deliberate deception attacks. As argued, the latter is often not covered by standard PIR and FHE guarantees and schemes should be enhanced by verification mechanisms to allow for correctness checks as well. We gave formal definitions of verifiable PIR and FHE protocols and detailed special features and variants of the baseline schemes examined in the literature. Furthermore, we outlined different existing vPIR and vFHE schemes to build an intuition for realizing verifiability in practice.

Based on this background knowledge, we turned to the main contribution of this thesis: Developing a construction for vPIR schemes starting from vFHE. It borrows the ideas of a similar conversion for the standard, non-verifiable setup but already improves upon it by operating on vectors instead of single data bits. We supported our reformulation of the PIR-from-FHE construction for matrices and vectors by precise descriptions of the resulting PIR protocols and by illustrative diagrams. To investigate whether the verifiability property of vFHE schemes transfers to the resulting PIR scheme, we proceeded in two stages. First, we concretized our definition of vFHE by settling on SNARG systems as the tool for verification and by embedding commitment schemes to bind computing parties to their internal data. Second, we performed the vector version of the PIR-from-FHE construction on the vFHE scheme obtained and stated the resulting vPIR protocol in detail. We then gave rigorous proofs that this vPIR protocol indeed exhibits all desirable properties of verifiable PIR, namely correctness, integrity and malicious privacy. To the best of our knowledge, we are the first to extend the FHE to PIR conversion of [66] set in the semi-honest threat model to the malicious scenario that introduces the additional property of verifiability. In particular, this allows to construct data retrieval schemes that ensure both the privacy of queries and the verifiability of results. These schemes are therefore suitable for settings where the presence of adversarial parties cannot be ruled out and might jeopardize the client's privacy and trust in the service provider.

In summary, our own work and main contributions of this thesis are the following. First, it serves as an approachable introduction to the topics of vPIR and vFHE and in particular presents overview tables of some of the most notable schemes designed in this context in Subsections 3.3 and 4.3. Moreover, we extend an existing construction of PIR from FHE to matrices (see Subsection 5.2) and vectors (see Subsection 5.3). As our main result, we then adapt the PIR-from-FHE construction to include the property of verifiability, detailed in Section 6. Wherever suitable, we visualize concepts by means of diagrams and interactive protocols. Unless otherwise stated, we designed and created them ourselves.

Drawing on our results, we want to give an outlook for future work on the topic of vPIR from vFHE. First, recall that the use of SNARG-based verification was justified by their natural support of two-party computation and strong security guarantees. However, compared to other verification mechanisms, they are heavy machinery as they rely on strong assumptions that cannot be realized in the standard model. Therefore, it would

be desirable to base the vPIR-from-vFHE construction on more lightweight components. Considering the taxonomy of vFHE schemes, a promising approach could be to adapt HMACs to handle inputs from multiple parties. HMACs, or the more general notion of *Homomorphic Authenticators (HAs)*, are typically symmetric-key encryption schemes, i.e., the same key is used for creating and verifying authentication tags. The generation of these tags is typically embedded in the encryption algorithm. Clearly, it is not an option to hand the secret key to the server so it can encrypt and authenticate its inputs itself. Hence, alternatives have to be sought. Recently, Lu et al. proposed a *Multi-key Verifiable HE (MVHE)* scheme [48]. As the name suggests, inputs from multiple parties are supported by allowing for computations on ciphertexts encrypted under different keys. Their method combines regular multi-key HE schemes and *Multi-key Homomorphic Encrypted Authenticator (MHEA)* schemes. MHEAs extend multi-key HAs by an additional privacy requirement, ensuring that authenticators do not reveal (partial) information about the underlying data. MHEA is a new primitive the authors introduce in their paper for the first time. Inspired by their work, we envision the following strategy of turning a FHE protocol into a vFHE one that is based on HAs and can handle inputs from two parties.

- Compile standard single-party, single-key FHE into two-party, multi-key FHE (see [47]).
- Use a MHEA protocol (see [48]).
- Combine multi-key FHE and MHEA to obtain a MVHE scheme (see [48]).
- Show that our vPIR-from-vFHE construction applies to the resulting MVHE scheme.

Currently, the bottleneck of this procedure seems to lie in the existence of MHEA schemes. Even though the authors of [48] provide one concrete instantiation, it has the drawback of relying on a compiler, turning single-key HAs into their multi-key variants (see [28]), that leverages SNARK proofs. Hence, utilizing their MHEA scheme would not allow to circumvent SNARKs and would only complicate our construction.
Another question of interest is whether our proposal of strongly vFHE in Definition 6.1 can be made less interactive. Usually, clients delegating the evaluation of functions by means of vFHE protocols merely have to send the function and the encrypted inputs to a server that then performs the homomorphic computation and returns an encrypted response. To lay the foundation for the digest generation and the verification step essential to vPIR protocols, we included commitment schemes. Now, however, an additional round of interaction is required as the commitment has to be generated and shared prior to the actual function evaluation. Moreover, it is crucial that the client stores an accepted commitment locally, to guarantee its validity for the final response verification. It is because of this assurance that we question whether applying the Fiat-Shamir transformation, the standard way of turning interactive protocols into non-interactive ones, serves the purpose in this context.
We also leave it to future works to test a practical instantiation of our vPIR-from-vFHE construction. In this respect, we view our contribution as a cornerstone for subsequent research: While we transported the existing construction of PIR from FHE of [66] to the more recent state of the art by integrating the property of verifiability, it opens up further interesting questions that require closer investigation.

References

[1] G. Alagic et al. “Quantum fully homomorphic encryption with verification”. In: *International Conference on the Theory and Application of Cryptology and Information Security*. Springer. 2017, pp. 438–467.

[2] F. Armknecht et al. *A Guide to Fully Homomorphic Encryption*. Cryptology ePrint Archive, Paper 2015/1192. 2015. URL: https://eprint.iacr.org/2015/1192.

[3] A. Beimel. “Private information retrieval: A primer”. In: *Department of Computer Science, Ben-Gurion University (Submitted for Publication)* (2008).

[4] A. Beimel and Y. Stahl. “Robust information-theoretic private information retrieval”. In: *International Conference on Security in Communication Networks*. Springer. 2002, pp. 326–341.

[5] M. Bellare, R. Canetti, and H. Krawczyk. “Keying hash functions for message authentication”. In: *Annual international cryptology conference*. Springer. 1996, pp. 1–15.

[6] M. Bellare and P. Rogaway. “Introduction to modern cryptography”. In: *Ucsd Cse* 207 (2005).

[7] S. Ben-David, Y. T. Kalai, and O. Paneth. “Verifiable private information retrieval”. In: *Theory of Cryptography Conference*. Springer. 2022, pp. 3–32.

[8] N. Bitansky et al. *Recursive Composition and Bootstrapping for SNARKs and Proof-Carrying Data*. Cryptology ePrint Archive, Paper 2012/095. 2012. URL: https://eprint.iacr.org/2012/095.

[9] N. Bitansky et al. “Succinct non-interactive arguments via linear interactive proofs”. In: *Theory of Cryptography Conference*. Springer. 2013, pp. 315–333.

[10] A. Bois et al. “Flexible and Efficient Verifiable Computation on Encrypted Data”. In: *Public-Key Cryptography – PKC 2021*. Cham: Springer International Publishing, 2021, pp. 528–558. ISBN: 978-3-030-75248-4.

[11] Z. Brakerski, C. Gentry, and V. Vaikuntanathan. “(Leveled) fully homomorphic encryption without bootstrapping”. In: *ACM Transactions on Computation Theory (TOCT)* 6.3 (2014), pp. 1–36.

[12] V. Buterin. *Quadratic Arithmetic Programs: From Zero to Hero*. https://medium.com/@VitalikButerin/quadratic-arithmetic-programs-from-zero-to-hero-f6d558cea649. Accessed: 2026-02-06. 2016.

[13] L. de Castro and K. Lee. *VeriSimplePIR: Verifiability in SimplePIR at No Online Cost for Honest Servers*. Cryptology ePrint Archive, Paper 2024/341. 2024. URL: https://eprint.iacr.org/2024/341.

[14] S. Chatel et al. “Veritas: Plaintext encoders for practical verifiable homomorphic encryption”. In: *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*. 2024, pp. 2520–2534.

[15] M. Checri et al. *Achieving ”beyond CCA1” security for linearly homomorphic encryption, without SNARKs?* Cryptology ePrint Archive, Paper 2025/894. 2025. URL: https://eprint.iacr.org/2025/894.

[16] B. Chor et al. “Private information retrieval”. In: *Journal of the ACM (JACM)* 45.6 (1998), pp. 965–981.

[17] S. Colombo et al. “Authenticated private information retrieval”. In: *32nd USENIX security symposium (USENIX Security 23)*. 2023, pp. 3835–3851.

[18] S. Cook. “The P versus NP problem”. In: *Clay Mathematics Institute* 2.6 (2000), p. 3.

[19] I. Damgård. “Commitment schemes and zero-knowledge protocols”. In: *School organized by the European Educational Forum*. Springer. 1998, pp. 63–86.

[20] I. Damgård and E. Fujisaki. “A statistically-hiding integer commitment scheme based on groups with hidden order”. In: *International Conference on the Theory and Application of Cryptology and Information Security*. Springer. 2002, pp. 125–142.

[21] A. Das, S. Dutta, and A. Adhikari. “Indistinguishability against chosen ciphertext verification attack revisited: The complete picture”. In: *International Conference on Provable Security*. Springer. 2013, pp. 104–120.

[22] C. Devet, I. Goldberg, and N. Heninger. “Optimally robust private information retrieval”. In: *21st USENIX Security Symposium (USENIX Security 12)*. 2012, pp. 269–283.

[23] M. Dietz and S. Tessaro. “Fully Malicious Authenticated PIR”. In: *Advances in Cryptology – CRYPTO 2024*. Cham: Springer Nature Switzerland, 2024, pp. 113–147. ISBN: 978-3-031-68400-5.

[24] D. Evans, V. Kolesnikov, and M. Rosulek. “A Pragmatic Introduction to Secure Multi-Party Computation”. In: *Foundations and Trends in Privacy and Security* 2 (2018), pp. 70–246. ISSN: 2474-1558. URL: https://doi.org/10.1561/3300000019.

[25] B. Falk, P. Mishra, and M. Shtepel. “Malicious security for PIR (almost) for free”. In: *Annual International Cryptology Conference*. Springer Nature Switzerland, 2025, pp. 175–210.

[26] J. Fan and F. Vercauteren. *Somewhat Practical Fully Homomorphic Encryption*. Cryptology ePrint Archive, Paper 2012/144. 2012. URL: https://eprint.iacr.org/2012/144.

[27] P. Fauzi, M. N. Hovd, and H. Raddum. “On the IND-CCA1 Security of FHE Schemes”. In: *Cryptography* 6.1 (2022). ISSN: 2410-387X. URL: https://www.mdpi.com/2410-387X/6/1/13.

[28] D. Fiore and E. Pagnin. “Matrioska: A compiler for multi-key homomorphic signatures”. In: *International Conference on Security and Cryptography for Networks*. Springer. 2018, pp. 43–62.

[29] R. Gennaro, C. Gentry, and B. Parno. “Non-interactive Verifiable Computing: Outsourcing Computation to Untrusted Workers”. In: *Advances in Cryptology – CRYPTO 2010*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 465–482. ISBN: 978-3-642-14623-7.

[30] R. Gennaro and D. Wichs. "Fully homomorphic message authenticators". In: *International Conference on the Theory and Application of Cryptology and Information Security*. Springer. 2013, pp. 301–320.

[31] C. Gentry. *A fully homomorphic encryption scheme*. Stanford university, 2009.

[32] C. Gentry and D. Wichs. "Separating succinct non-interactive arguments from all falsifiable assumptions". In: *Proceedings of the forty-third annual ACM symposium on Theory of computing*. 2011, pp. 99–108.

[33] O. Goldreich. "Encryption Schemes". In: *Foundations of Cryptography*. Cambridge University Press, 2004, pp. 373–496.

[34] O. Goldreich. *Foundations of Cryptography: Volume 2, Basic Applications*. 1st. USA: Cambridge University Press, 2009. ISBN: 052111991X.

[35] O. Goldreich. "General Cryptographic Protocols". In: *Foundations of Cryptography*. Cambridge University Press, 2004, pp. 599–764.

[36] O. Goldreich. "Introduction". In: *Foundations of Cryptography*. Cambridge University Press, 2001, pp. 1–29.

[37] O. Goldreich. "Zero-Knowledge Proof Systems". In: *Foundations of Cryptography*. Cambridge University Press, 2001, pp. 184–330.

[38] J. Groth. "On the size of pairing-based non-interactive arguments". In: *Annual international conference on the theory and applications of cryptographic techniques*. Springer. 2016, pp. 305–326.

[39] R. Hayashi et al. "Multi-query verifiable PIR and its application". In: *International Conference on Cryptology and Network Security*. Springer. 2024, pp. 166–190.

[40] M.-Y. M. Huang et al. *Fully Homomorphic Encryption with Efficient Public Verification*. Cryptology ePrint Archive, Paper 2024/1764. 2024. URL: `https://eprint.iacr.org/2024/1764`.

[41] I. Ionescu and R. F. Olimid. *Commitment Schemes for Multi-Party Computation*. 2025. URL: `https://arxiv.org/abs/2506.10721`.

[42] J. Kim and A. Yun. "Secure fully homomorphic authenticated encryption". In: *IEEE Access* 9 (2021), pp. 107279–107297.

[43] C. Knabenhans et al. "vFHE: Verifiable Fully Homomorphic Encryption". In: *Proceedings of the 12th Workshop on Encrypted Computing & Applied Homomorphic Cryptography*. WAHC '24. Salt Lake City, UT, USA: Association for Computing Machinery, 2024, pp. 11–22. ISBN: 9798400712418. URL: `https://doi.org/10.1145/3689945.3694806`.

[44] S. Kumar Pandey, S. Sarkar, and M. Prasad Jhanwar. "Relaxing IND-CCA: Indistinguishability against chosen Ciphertext verification attack". In: *International Conference on Security, Privacy, and Applied Cryptography Engineering*. Springer. 2012, pp. 63–76.

[45] E. Kushilevitz and R. Ostrovsky. "Replication is not needed: Single database, computationally-private information retrieval". In: *Proceedings 38th annual symposium on foundations of computer science*. IEEE. 1997, pp. 364–373.

[46] Y. Lindell and B. Pinkas. “Secure two-party computation via cut-and-choose oblivious transfer”. In: *Journal of cryptology* 25.4 (2012), pp. 680–722.

[47] A. López-Alt, E. Tromer, and V. Vaikuntanathan. “Multikey fully homomorphic encryption and applications”. In: *SIAM Journal on Computing* 46.6 (2017), pp. 1827–1892.

[48] Y. Lu, K. Hara, and K. Tanaka. “Multikey verifiable homomorphic encryption”. In: *IEEE Access* 10 (2022), pp. 84761–84775.

[49] G. K. Mahato and S. K. Chakraborty. “A fast verifiable fully homomorphic encryption technique for secret computation on cloud data”. In: *International Journal of Information Technology* 17.4 (2025), pp. 2193–2207.

[50] M. Manulis and J. Nguyen. “Fully homomorphic encryption beyond IND-CCA1 security: Integrity through verifiability”. In: *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer. 2024, pp. 63–93.

[51] S. Micali, O. Goldreich, and A. Wigderson. “How to play any mental game”. In: *Proceedings of the Nineteenth ACM Symp. on Theory of Computing, STOC*. ACM New York. 1987, pp. 218–229.

[52] D. Natarajan et al. *CHEX-MIX: Combining homomorphic encryption with trusted execution environments for two-party oblivious inference in the cloud*. Cryptology ePrint Archive, Paper 2021/1603. 2021. URL: `https://eprint.iacr.org/2021/1603.pdf`.

[53] P. Paillier. “Public-key cryptosystems based on composite degree residuosity classes”. In: *International conference on the theory and applications of cryptographic techniques*. Springer. 1999, pp. 223–238.

[54] C. Peikert and S. Shiehian. “Noninteractive zero knowledge for NP from (plain) learning with errors”. In: *Annual International Cryptology Conference*. Springer. 2019, pp. 89–114.

[55] M. Rathee, K. Lee, and R. A. Popa. *Verifiable PIR with Small Client Storage*. Cryptology ePrint Archive, Paper 2025/1714. 2025. URL: `https://eprint.iacr.org/2025/1714`.

[56] M. Renard. “Contributions to FHE security against CCA adversaries”. PhD thesis. Université Paris-Saclay, 2025.

[57] M. H. Santriaji et al. “DataSeal: Ensuring the verifiability of private computation on encrypted data”. In: *2025 IEEE Symposium on Security and Privacy (SP)*. IEEE. 2025, pp. 2378–2394.

[58] H. Shen et al. “EAPIR: Efficient and Authenticated Private Information Retrieval with Fast Server Processing”. In: *Australasian Conference on Information Security and Privacy*. Springer. 2025, pp. 391–398.

[59] Technische Universität München. *TUM Citation Guide*. `https://mediatum.ub.tum.de/doc/1236069/1236069.pdf`. Accessed: 2026-03-06. 2025.

[60] A. Viand, C. Knabenhans, and A. Hithnawi. "Verifiable Fully Homomorphic Encryption". In: *arXiv preprint arXiv:2301.07041* (2023). URL: https://arxiv.org/abs/2301.07041.

[61] X. Wang and L. Zhao. "Verifiable Single-Server Private Information Retrieval". In: *Information and Communications Security*. Cham: Springer International Publishing, 2018, pp. 478–493.

[62] Y. Wang et al. *Crust: Verifiable and Efficient Private Information Retrieval With Sublinear Online Time*. Cryptology ePrint Archive, Paper 2023/1607. 2023. URL: https://eprint.iacr.org/2023/1607.

[63] A. El-Yahyaoui and M. D. Ech-Cherif el Kettani. "A verifiable fully homomorphic encryption scheme for cloud computing security". In: *Technologies* 7.1 (2019), p. 21.

[64] R. Yang, Z. Yu, and W. Susilo. "Fully Homomorphic Encryption with Chosen-Ciphertext Security from LWE". In: *Annual International Cryptology Conference*. Springer. 2025, pp. 371–405.

[65] A. C.-C. Yao. "How to generate and exchange secrets". In: *27th annual symposium on foundations of computer science (Sfcs 1986)*. IEEE. 1986, pp. 162–167.

[66] X. Yi et al. "Single-database private information retrieval from fully homomorphic encryption". In: *IEEE Transactions on Knowledge and Data Engineering* 25.5 (2012), pp. 1125–1134.

[67] L. F. Zhang and R. Safavi-Naini. "Verifiable Multi-server Private Information Retrieval". In: *Applied Cryptography and Network Security*. Cham: Springer International Publishing, 2014, pp. 62–79. ISBN: 978-3-319-07536-5.

[68] W. Zhang et al. "VPIR: An efficient verifiable private information retrieval scheme resisting malicious cloud server". In: *Telecommunication Systems* 86.4 (2024), pp. 743–755.

[69] X. Zhang et al. "Phalanx: An FHE-Friendly SNARK for Verifiable Computation on Encrypted Data". In: *Proceedings of the 2025 ACM SIGSAC Conference on Computer and Communications Security*. 2025, pp. 4035–4048.

[70] L. Zhao, X. Wang, and X. Huang. "Verifiable single-server private information retrieval from LWE with binary errors". In: *Information Sciences* 546 (2021), pp. 897–923. ISSN: 0020-0255. URL: https://doi.org/10.1016/j.ins.2020.08.071.

[71] F. Zhou et al. "Efficient private information retrievals for single-server based on verifiable homomorphic encryption". In: *Computer Standards & Interfaces* 93 (2025), p. 103961.