Privacy-Preserving Pattern Matching on Encrypted Data

Anis Bkakria[1], Nora Cuppens[1,2], and Frédéric Cuppens[1,2]

[1] IMT Atlantique, Rennes, France
[2] Polytechnique Montréal, Montréal, Canada

Abstract. Pattern matching is one of the most fundamental and important paradigms in digital forensics and cyber threat intelligence approaches. While it is a straightforward operation when performed on plaintext data, it becomes a challenging task when the privacy of both the analyzed data and the analysis patterns must be preserved. In this paper, we propose P^{3}MED: a new provably correct, secure, and quite efficient construction that allows arbitrary pattern matching over encrypted data while fully protecting both the data to be analyzed and the patterns to be matched. That is, the entity that will perform pattern matching will learn nothing about the patterns to be searched as well as the data to be inspected, except the presence or the absence of a set of "unknown" patterns (the entity charged to perform pattern matching will not have access to the analysis patterns plaintexts). Compared to existing solutions, the construction we propose has some remarkable properties: (1) the size of the ciphertext is linear to the size of plaintext and independent of the sizes and the number of the analysis patterns; (2) the sizes of the issued trapdoors are constant on the size of the data to be analyzed; and (3) the search complexity is linear to the size of the data to be inspected and is constant on the sizes of the analysis patterns. The conducted evaluations show that P^{3}MED drastically improves the performance of the most efficient state of the art solution.

Keywords: Searchable encryption · Pattern Matching · Privacy-preserving threat intelligence.

1 Introduction

In recent years, the cyber attacks on Sony in 2012, the German Parliament in 2015, the US Democratic National Committee and Bangladesh National Bank in 2016, along with the repeated attacks on Ukraine's ministries and infrastructures have strongly demonstrate that cybercrimes can severely compromise the activities and reputations of any private and public sector entities in both the developing and developed world. These cyber attacks are sophisticated, large scale, and multi-vector in the sense that they can target simultaneously, computers, smartphones, connected objects, and even industrial systems. The number of enterprises and organizations affected by cyber attacks is increasing exponentially with a financial impacts of more and more cumbersome. This is due to the

fact that today, the speed of evolution of cyber attacks and their sophistication exceeds by far the rate of evolution of cyber security solutions.

To cope effectively with these cybercrime threats, it is necessary to define new cybersecurity techniques that rely mainly on the collaboration between different involved entities. These entities include in one side companies, organizations, and individuals that are targeted by cyber attacks, and in the other side, security editors that are in charge of defining and providing strategies for effectively detect and prevent cyber attacks. The overall objective is to build a threat intelligence platform allowing the sharing and correlating Indicators of Compromise of targeted attacks, financial fraud information, vulnerability information (e.g., the platform defined in [20]). To be useful, such platform should, in one side, be fueled by data owners, i.e., companies, organizations, and individuals that agree to share the traces (e.g., network and operating system traces) of the cyber attacks that they have suffered. In the other side, the platform should allow the security editors to analyze (e.g., search specific patterns) and correlate the traces that are shared by the data owners. The considered threat intelligence platform is often managed by non-fully trusted third-party service provider (SP) which provides the required storage space and computation power with affordable cost.

Unfortunately, both data owners (i.e., attack traces owners) and security editors are still very reluctant for adopting such kind of threat intelligence platforms because of two main reasons. First, the traces to be shared contain often highly sensitive information that may raise serious security and/or business threats when disclosed to non-fully trusted third parties (e.g, SP). Second, the shared traces analysis rely mainly on techniques that use pattern matching for inspecting and detecting malicious behaviors. Those analysis patterns are the result of extensive threat intelligence conducted by security editors. They are often put forward as a key competitive differentiator arguing that they can cover a wider set of malicious behaviors. Thus, security editors are typically reluctant to share their analysis patterns with non-fully trusted third-parties.

Existing solutions that may be used to overcome the previous two limitations rely mainly on searchable encryption based techniques [7, 5, 3, 1, 4, 2, 6]. Unfortunately, these techniques suffer from at least one of the following limitations. First, the lack of support for pattern-matching with evolving patterns, such as virus signatures which are updated frequently (case of symmetric searchable encryption [5, 3, 4, 2]); second, the lack of support for variable pattern lengths (e.g., tokenization-based techniques such as BlindBox [6]); third, the incompleteness of pattern detection methods which yield false negatives (case of BlindIDS [7]); and fourth, the disclosure of detection patterns (case of searchable encryption with shiftable trapdoors [1]). We provide a full comparison with related literature in Section 2.

In this paper, we propose P^{3}MED: a technically sound construction that supports pattern matching over encrypted data for adaptively chosen and variable lengths patterns. P^{3}MED is (1) market compliant meaning that the third-party SP will learn nothing about the patterns that will be used by security editors for analyzing the shared traces, and (2) privacy-friendly, signifying that the third-

party SP will learn nothing about the shared traces except the presence or the absence of a set of unknown analysis patterns (i.e., SP will not have access to security editors' analysis patterns plaintexts). From practical point of view, P^{3}MED has some remarkable properties. First, the size of the ciphertext depends only on the size of the plaintext (it is independent of the sizes and the number of analysis patterns). Second, the size of the issued trapdoors is independent of the size of the data to be analyzed. Third, the search complexity depends only on the size of the data to be analyzed and is constant on the size of the analysis patterns. Our construction is – to our knowledge – the first construction to provide all previously mentioned properties without using costly and complex cryptographic scheme such as fully homomorphic encryption. Both theoretical complexity and experimental evaluations demonstrate that our construction improves drastically the performance of the most efficient state of the art solution SEST [1].

The paper is organized as follows. Section 2 reviews related work and details the main contributions of our work. Section 3 presents the architecture of our solution and the main security and privacy requirements that it aims to achieve. Section 4 introduces the construction we propose. In Section 5, we describe the assumption used by our scheme to achieve provable security, and provide security proofs. In Sections 6 and 7, we discuss the complexity of our construction and provide an empirical evaluation. Finally, section 8 concludes.

2 Related Work

One possible solution for pattern matching over encrypted traffic is to use techniques that allow functions evaluation over encrypted data. Generic approaches such as fully homomorphic encryption (FHE) [8, 10] and functional encryption (FE) [9] are currently impractical due to their very high complexities.

Several searchable encryption (SE) techniques have been proposed for keyword searching over encrypted data [5, 3, 4, 2]. The main idea is to associate a trapdoor with each keyword to allow searching for these keywords within a given encrypted data. Ideally, some entity which does not have access to the plaintext and encryption key should learn nothing about the plaintext except the presence or the absence of the keyword. For most existing SE techniques, searches are performed on keywords that have been pre-chosen by the entity encrypting the data. Such approaches are more suitable for specific types of searches, such as database searches in which records are already indexed by keywords, or in the case of emails filtering in which flags such as "urgent" are used. Unfortunately, SE techniques become useless when the set of keywords cannot be known before encryption. This is usually the case for messaging application and Internet browsing traffic where keywords can include expressions that are not sequences of words *per se* (e.g., /chamjavanv.inf?aapf/login.jsp?=). Our construction offers better search flexibility as, even after the plaintext has been encrypted and sent, it can allow arbitrarily-chosen keywords searching without re-encryption.

To overcome the previous limitations, tokenization-based approaches have been proposed. In [6], the authors propose BlindBox, an approach that splits the data to be encrypted into fragments of the same size l and encrypts each of those fragments using a searchable encryption scheme where each fragment will represent a keyword. Nevertheless, this solution suffers from two limitations: (1) it is useful only if all the searchable keywords have the same length l. obviously the previous condition is seldom satisfied in real-world threat intelligence applications. If we want to use this approach with keyword of different lengths $\mathcal{L}$, we should for each $l_i \in \mathcal{L}$, split the data to be encrypted into fragments of size l_i and encrypt them, which quickly becomes bulky. (2) The proposed approach may easily cause false negatives since, even if the keyword is of size l (the size of each fragment), it cannot be detected if it straddles two fragments. Recently, In [7], Canard et al. proposed BlindIDS – a public key variant of the BlindBox approach [6] that additionally ensures keywords indistinguishability. That is, the entity that is going to search over the encrypted data will lean nothing about the keywords. Unfortunately, BlindIDS suffers from the same limitations as BlindBox. Our construction addresses the main drawbacks of these tokenization-based techniques since it allows for patterns of arbitrary (but bounded) sizes to be matched against the encrypted data, without false negatives or false positives.

Several approaches [11–13] proposed solutions for substring search over encrypted data based on secure multi-party computation. Unfortunately, these solutions require often several interactions between the searcher and the data encrypter. In addition, a part of the computations required for performing the pattern matching operations will be carried out by the data owner.

As pointed out in [1], anonymous predicate encryption (e.g., [14]) or hidden vector encryption [15] may provide a convenient solution for pattern matching over encrypted stream. However, in order to search a pattern p of length l on a data of length n, the searcher should obtain $n - l$ keys to be able to check the presence of p on every possible offset of the data, which is clearly a problem when dealing with large datasets.

One of the most interesting techniques for pattern matching over encrypted traffic is the searchable encryption with shiftable trapdoor (SEST) [1]. The proposed construction relies on public-key encryption and bilinear pairings to overcome most of the limitations of previously mentioned techniques. It allows patterns of arbitrary lengths to be matched against the encrypted data, without false negatives or false positives. This improvement comes at the cost of the practicability of the technique. In fact, the proposed schema requires a public key of size linear to the size of the data to be encrypted (a public key of $\simeq 8000$ GB is required for analyzing 1GB of data). Moreover, the trapdoor generation technique used by the SEST leaks many information (such as, the number of different characters as well as the maximum number of occurrences of a character) about the patterns to be searched. Furthermore, the number of pairings needed for testing the presence of a keyword in an offset of the data depends on the maximum number of occurrences of the characters contained in the pattern. This makes the proposed technique quite inefficient when used for bit level

matching. By contrast, to test the presence of a pattern in an encrypted data, P^{3}MED requires a constant number of pairings in the size of the pattern (see Section 6 for more details). This makes P^{3}MED more efficient when matching long pattern at bit level.

As we have seen, many different approaches can be used to address pattern matching over encrypted data. To give better understanding of the benefits of our approach compared to existing ones, we provide in Table 1 a comparative overview of their asymptotic complexities, and their ability to ensure the security properties we are aiming to provide. Note that we only consider BlindBox (a symmetric searchable encryption-based solution), BlindIDS (an asymmetric searchable encryption-based solution), Predicate Encryption/Hidden Vector Encryption and the SEST approach. Other approaches, as explained before, require data re-encryption each time a new keyword is considered [5, 3, 4, 2], induce higher complexity [9, 8, 10], require interactivity [11–13] or ensure weaker privacy level [5].

	Primitives				
	BlindBox	BlindIDS	PE/HVE	SEST	P^{3}MED
Number of Trapdoors	$O(s \cdot q)$	$O(q)$	$O(n \cdot q)$	$O(q)$	$O(q)$
Public Parameters size	$O(1)$	$O(1)$	$O(n)$	$O(n)$	$O(l_i)$
Ciphertext size	$O(n \cdot L)$	$O(n \cdot L)$	$O(n)$	$O(n)$	$O(n)$
Trapdoors size	$O(q)$	$O(q)$	$O(n \cdot q)$	$O(q)$	$O(q)$
Search complexity	$q \cdot \log(q)$ comparisons	q pairings	$q \cdot n$ pairings	$2 \times \prod_1^q l_i \cdot n$ pairings	$2 \cdot q \cdot n$ pairings
Arbitrary trapdoors	✗	✗	✓	✓	(✓)
Trapdoor's privacy	✗	(✓)	✗	✗	✓
Correctness (false positives)	✗	✗	✓	✓	✓

Table 1. Complexity and ensured security properties comparison between related work and P^{3}MED. The scalars n, q, l_i, L, s denote respectively the length of the data to be analyzed, the number of issued trapdoors, the length of each issued trapdoor, the number of different lengths among the q trapdoors and the number of data encrypters.

According to the Table 1, P^{3}MED is the only primitive that simultaneously enables arbitrary trapdoors (with upper bounded pattern size), provides a correct pattern detection, and ensures the privacy of the used trapdoors. Thanks to the data fragmentation approach (Section 4) we use, our construction does not require either very large public parameters or very large encryption key as SEST and PE/HVE. Moreover, the search complexity of our construction is lower than SEST by a factor of l_i (the length of the issued trapdoor), since it is constant in the size of the trapdoors. Therefore, P^{3}MED is an interesting middle way which almost provides the best of PE/HVE and SEST while ensuring the trapdoors privacy. Its only drawback compared to PE/HVE and SEST is the upper

bounded size of trapdoors that should be fixed before the data encryption which we believe to be a reasonable price to pay to achieve all the other features.

3 Architecture and Security

3.1 Considered Architecture

In this section, we describe the main architecture of our approach. It involves three main parties: the data owner (or data holder), the Security Editor, and the Service Provider. The data owner (DO) represents the entities that hold data containing cyber attack traces and want to outsource them to the threat intelligence platform. The Security Editor (SE) is the entity that is supposed to analyze the outsourced traces to define new attack detection signatures. SE's analysis queries will probably contain specific patterns used to detect malicious behaviors. These analysis pattern are very valuable assets for the SE as they require continuous and extensive threat intelligence processes to be expressed and maintained. Finally, Service Provider (SP) is stakeholder that hosts the threat intelligence platform and offers computation infrastructures that will be used for executing different pattern matching queries performed by SE. SP is often partner with SE in order to implement and perform the analysis logics.

3.2 Security Requirements and Considered Hypothesis

Before presenting the formal description of our construction, we first present the hypothesis we are considered and the security properties that we aim to provide.

SE is considered in P^{3}MED as an *Honest-but-curious* entity. That is we expect SE to issue analysis queries that contain true malicious patterns that are mainly designed to detect attacks. This is a fairly reasonable assumption since, in our construction, the DO can check and authorize (by issuing a trapdoor) the analysis patterns that will be searched over the his/her outsourced data. Thus, SE will not defile its reputation by issuing incorrect or misleading analysis patterns. Nevertheless, we expect the SE to be curious: it may try to derive information about the analyzed data by accessing the (encrypted) data hosted by the SP or/and by correlating the analyses results. Hypothesis 1 and Requirement 1 summarize the previous considerations.

Hypothesis 1 *SE will perform queries that are only designed to analyze the outsourced cyber attak traces.*

Requirement 1 *SE should learn nothing about the analyzed traces except the parts that entirely match at least one of the used analysis patterns.*

SP is also considered in our construction as an *Honest-but-Curious* entity. That is, we suppose that it will honestly perform the pattern matching operations required by the analysis query issued by the SE. However, we suppose that the SP may try to learn additional information about either or both of the DO's

outsourced traces and the analysis patterns used by the SE. In addition, we assume that the SP may try to create values by analyzing other third-parties traces using the set of patterns used by the SE for the analysis of DO's outsourced traces. We then consider the following hypothesis and security requirements.

Hypothesis 2 *SP will correctly perform the pattern matching operations required by the SE's analysis queries and will correctly report analysis results to the SE.*

Requirement 2 *SP should learn nothing about the analyzed traces except the indexes in which the analysis patterns fully match the (a part of) traces.*

Requirement 3 *SP should learn nothing about the analysis patterns (queries) performed by the SE except the indexes in which the patterns fully match (a part of) the traces.*

Requirement 4 *The patterns used by the SE for the analysis of a DO's traces should be useless for analyzing any other third party's traces.*

In addition, we consider DO to be an honest entity. That is, DO will not try to disclose the set of SE's patterns used to analyze its data. Moreover, we suppose that the SP and SE will not collude to learn more information about the traffic. We believe that this last assumption is fairly reasonable since, in a free market environment, an open dishonest behavior will result in considerable damages for both entities.

Hypothesis 3 *SP and SE will not collude to gain more information about the DO's outsourced traces.*

Finally, we require our construction to provide a correct analysis results. That is, (1) any malicious traces (the traffic that matches at least one of the SE's analysis patterns when not encrypted) must be detected by our construction (no false negatives), and (2) we require that for any safe traffic (the traffic that does not match any of the SE's analysis patterns when not encrypted), the probability that our construction returns a false positive is negligible.

Requirement 5 *The proposed construction should provide a correct pattern matching results.*

3.3 Syntax of the proposed construction

P^3MED is defined by five algorithms that we denote **Setup**, **Keygen**, **Encrypt**, **Issue**, and **Test**. The first three algorithms are performed by the DO. The **Issue** algorithm is performed collaboratively by the DO and the SE. The **Test** algorithm is performed by the SP.

- **Setup**$(1^\lambda, p_{max})$ is a probabilistic algorithm that takes as input a security parameter λ as well as the largest analysis pattern size p_{max} (i.e., the size of largest pattern that can be matched by our construction). It returns the public parameters *params* which will be an implicit input to all following algorithms.
- **Keygen**(Σ) is a probabilistic key generation algorithm that takes as input a finite set of symbols Σ representing the alphabet to be used for encoding the data to be analyzed as well as the analysis patterns to be searched. It outputs a secret key K.
- **Encrypt**$(K, \mathcal{B})$ is a probabilistic algorithm that takes as input a finite sequence (String) of symbols $\mathcal{B}$ of Σ and the secret key K. it returns a ciphertext C.
- **Issue**(K, w) is a probabilistic algorithm that takes as input the secret key K, and w – a finite sequence of elements of Σ – and returns a trapdoor td_w.
- **Test**(C, td_w) is a deterministic algorithm that takes as input a ciphertext C encrypting a sequence of n elements $\mathcal{B} = \sigma_0 \cdots \sigma_{n-1}$ of Σ, and the trapdoor td_w issued for the sequence of l Σ's elements $w = \sigma_{w,0} \cdots \sigma_{w,l-1}$ $(l \leq p_{max})$. The algorithm returns the set of indexes $\mathcal{I} \subset \{0, n-l-1\}$ where for each $i \in \mathcal{I}$, $\sigma_i \cdots \sigma_{i+l-1} = \sigma_{w,0} \cdots \sigma_{w,l-1}$.

We note that the sizes of the elements defined in the previous algorithms, i.e., the size of the data to be analyzed $\mathcal{B}$, the size of the pattern to be searched w, and the largest analysis pattern size p_{max} refer to the number of symbols of Σ that compose each element. In addition, we note that P^{3}MED does not consider a decryption algorithm since there is no need for decrypting the outsourced traces. However, we stress that a decryption feature can be straightforwardly performed by issuing a trapdoor for all characters $\sigma \in \Sigma$ and running the Test algorithm on the encrypted data for each of them.

3.4 Model for security requirements

As said in Section 3.2, there are mainly 4 security requirements that should be satisfied by our construction; Trace indistinguishability (Requirements 1 and 2), pattern indistinguishability (Requirement 3), pattern usefulness (Requirement 4), and the correctness property (Requirement 5).

In the following definition, we use the game-based security definition proposed in [1] to define the trace indistinguishability requirement by adapting the standard notion of IND-CPA which requires that no adversary $\mathcal{A}$ (e.g., the SE or SP), even with an access to an oracle $\mathcal{O}^{iss}$ that issues a trapdoor td_{p_i} for any adaptively chosen pattern p_i, can decide whether an encrypted trace encrypts T_0 or T_1 as long as the trapdoors $\{td_{p_i}\}$ issued by $\mathcal{O}^{iss}$ do not allow trivial distinction of the traces T_0 and T_1.

Definition 1 (Trace indistinguishability). *Let λ be the security parameter, Σ be the alphabet to be used, $\mathcal{A}$ be the adversary and $\mathcal{C}$ be the challenger. We consider the following game that we denote $Exp^{T\_IND\_CPA}_{\mathcal{A},\beta}$:*

1. *Setup:* $\mathcal{C}$ *executes* $Setup(1^\lambda, p_{max})$ *to generate params and the algorithm* $Keygen(\Sigma)$ *to generate the secret key* K*. Then it sends params to the adversary.*
2. *Query:* $\mathcal{A}$ *can adaptively ask* $\mathcal{O}^{iss}$ *to issue a trapdoor* td_{p_i} *for any pattern* $p_i = \sigma_{i,0} \cdots \sigma_{i,m-1}$ *where* $\sigma_{i,j} \in \Sigma$*. We denote* $\mathcal{P}$ *the set of patterns submitted by* $\mathcal{A}$ *to* $\mathcal{O}^{iss}$ *in this phase.*
3. *Challenge: Once* $\mathcal{A}$ *decides that Phase (2) is over, it chooses two traces* $T_0 = \sigma^*_{0,0} \cdots \sigma^*_{0,m-1}$ *and* $T_1 = \sigma^*_{1,0} \cdots \sigma^*_{1,m-1}$ *and sends them to* $\mathcal{C}$*.*
   (a) *If* $\exists p = \sigma_0 \cdots \sigma_l \in \mathcal{P}$*,* $k \in \{0,1\}$*, and* j *such that:*

$$\sigma^*_{k,j} \cdots \sigma^*_{k,j+l} = \sigma_0 \cdots \sigma_l \neq \sigma^*_{1-k,j} \cdots \sigma^*_{1-k,j+l}$$ *then return 0.*

   (b) $\mathcal{C}$ *chooses a random* $\beta \in \{0,1\}$*, creates* $C = Encrypt(K, T_\beta)$*, and sends it to* $\mathcal{A}$*.*
4. *Guess.* $\mathcal{A}$ *outputs the guess* β'*.*
5. *Return* $(\beta = \beta')$*.*

We define $\mathcal{A}$*'s advantage by* $Adv^{Exp^{T\text{-}IND\text{-}CPA}_{\mathcal{A},\beta}}(\lambda) = |Pr[\beta = \beta'] - 1/2|$*.* P^3MED *is said to be trace indistinguishable if* $Adv^{Exp^{T\text{-}IND\text{-}CPA}_{\mathcal{A},\beta}}(\lambda)$ *is negligible.*

We note that in the previous definition, the restriction used in phase (3)(a) ensures that if one of the traces T_k ($k \in \{0,1\}$) contains a pattern $p_i \in \mathcal{P}$ in the position j, then this is also the case for T_{1-k}. If such a restriction is not considered, $\mathcal{A}$ will trivially win the game by running $Test(C, td_{p_i})$.

We want to be able to evaluate the advantage of the SP for using the issued trapdoors to analyze other third-parties' traces (i.e., traces that are not provided and encrypted by the DO). Since encrypted traces and trapdoors should be created using the same secret key, such an advantage is equivalent to the ability of the SP to forge valid DO's encrypted traces.

Definition 2 (Encrypted Trace Unforgeability). *Let* λ *be a security parameter,* Σ *be the alphabet to be used,* $\mathcal{A}$ *be the adversary,* $\mathcal{O}^{iss}$ *be an oracle that issues a trapdoor for any adaptively chosen pattern, and* $\mathcal{O}^R$ *be an oracle that encrypts any adaptively chosen trace. We consider the following* $Exp^{ETF}_{\mathcal{A}}$ *game:*

1. *Setup:* $\mathcal{C}$ *executes* $Setup(1^\lambda, p_{max})$ *to generate params and the algorithm* $Keygen(\Sigma)$ *to generate the secret key* K*. Then it sends params to the adversary.*
2. *Query:*
   - $\mathcal{A}$ *can adaptively ask* $\mathcal{O}^{iss}$ *to issue a trapdoor* td_{p_i} *for any chosen pattern* $p_i = \sigma_{i,1} \cdots \sigma_{i,m}$ *where* $\sigma_{i,j} \in \Sigma$*. We denote* $\mathcal{P}$ *the set of patterns submitted by* $\mathcal{A}$ *to* $\mathcal{O}^{iss}$ *in this phase.*
   - $\mathcal{A}$ *can adaptively ask* $\mathcal{O}^R$ *to create* $C^T = Encrypt(K, T)$*. We denote* $\mathcal{T}$ *the set of traces encrypted by the* $\mathcal{O}^R$*.*
3. *Forgery: The adversary chooses the trace* $T^* \notin \mathcal{T}$ *such that* T^* *contains* p_t ($p_t \in \mathcal{P}$) *at index* i *and forges the encrypted traces* C^* *of* T^**.*

We define $\mathcal{A}$'s advantage of winning the game $Exp_{\mathcal{A}}^{ETF}$ by $Adv^{Exp_{\mathcal{A}}^{ETF}}(\lambda) = Pr[i \in Test(C^*, td_{p_t})]$. P^3MED is said to be encrypted trace forgery secure if $Adv^{Exp_{\mathcal{A}}^{ETF}}(\lambda)$ is negligible.

The following definition formalizes the patterns indistinguishability property. That is, we evaluate the advantage of the SP to decide whether a trapdoor encrypts the patterns w_0^* or w_1^* even with an access to an oracle $\mathcal{O}^{iss}$ that issues a trapdoor for any adaptively chosen pattern.

Definition 3 (Pattern Indistinguishability). *Let λ be the security parameter, Σ be the alphabet to be used, $\mathcal{A}$ be the adversary and $\mathcal{C}$ the challenger. We consider the following game that we denote $Exp_{\mathcal{A}}^{P\_IND\_CPA}$:*

1. *Setup: $\mathcal{C}$ executes $Setup(1^\lambda, p_{max})$ to generate params and the algorithm $Keygen(\Sigma)$ to generate the secret key K. Then it sends params to the adversary.*
2. *Observation: $\mathcal{A}$ may observe the ciphertext C^{T_i} of a set of (unknown) traces $T_i \in \mathcal{T}$.*
3. *Query: $\mathcal{A}$ can adaptively ask $\mathcal{O}^{iss}$ to issue a trapdoor td_{w_i} for any chosen pattern $w_i = \sigma_{i,1} \cdots \sigma_{i,l}$ where $\sigma_{i,j} \in \Sigma$. We denote by $\mathcal{P}$ the set of patterns submitted by $\mathcal{A}$ to $\mathcal{O}^{iss}$ in this phase.*
4. *Challenge: Once $\mathcal{A}$ decides that Phase (2) is over, it chooses two patterns $w_0^* = \sigma_{0,0}^* \cdots \sigma_{0,l}^*$ and $w_1^* = \sigma_{1,0}^* \cdots \sigma_{1,l}^*$ such that $w_0^*, w_1^* \notin \mathcal{P}$ and sends them to $\mathcal{C}$. If $\exists T \in \mathcal{T}$ such that $w_0^* \in T$ or $w_1^* \in T$ then return 0. Otherwise, $\mathcal{C}$ chooses a random $\beta \in \{0, 1\}$, creates $td_{w_\beta^*}$, and sends it to $\mathcal{A}$.*
5. *Guess:*
   - *$\mathcal{A}$ may try to forge the ciphertext of chosen traces and uses the Test algorithm to figure out the chosen value of β.*
   - *$\mathcal{A}$ outputs the guess β'.*
6. *Return ($\beta = \beta'$).*

We define the advantage of the adversary $\mathcal{A}$ for winning the game $Exp_{\mathcal{A},\beta}^{P\_IND\_CPA}$ by $Adv^{Exp_{\mathcal{A},\beta}^{P\_IND\_CPA}}(\lambda) = |Pr[\beta' = \beta] - 1/2|$. P^3MED is said to be pattern indistinguishable if $Adv^{Exp_{\mathcal{A},\beta}^{P\_IND\_CPA}}(\lambda)$ is negligible.

Definition 4 (Correctness). *Let $\mathcal{B} = \sigma_0, \cdots \sigma_{m-1}$ and $p = \sigma_{p,0}, \cdots \sigma_{p,l_t-1}$ be respectively the traces to be analyzed and the pattern to be searched. P^3MED is consistent iff the following conditions hold:*

1. *$Pr[i \in Test(Encrypt(\mathcal{B}, K), Issue(p, K))] = 1$ if $\mathcal{B}$ contains p at index i.*
2. *$Pr[i \in Test(Encrypt(\mathcal{B}, K), Issue(p, K))]$ is negligible if $\mathcal{B}$ does not contain p at index i.*

Condition (1) of the previous definition ensures that the Test algorithm used in our construction produces no false negatives. Condition (2) ensures that false positives (i.e., the case in which Test algorithm returns i notwithstanding the fact that $\sigma_i \cdots \sigma_{i+l-1} \neq \sigma_{w,0} \cdots \sigma_{w,l-1}$) only occur with negligible probability.

4 Construction

In this section, we give the intuition behind our construction. We briefly introduce bilinear maps and describe the way we use our construction to enable arbitrary pattern matching over encrypted traffic.

4.1 Bilinear Environment

Let $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ be three finite cyclic groups of large prime order p. We assume that there is an asymmetric bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ such that, for all $a, b \in \mathbb{Z}_p$ the following conditions hold:

- For all $g \in \mathbb{G}_1, \widetilde{g} \in \mathbb{G}_2$, $e(g^a, \widetilde{g}^b) = e(g, \widetilde{g})^{a \cdot b}$
- For all $g \in \mathbb{G}_1, \widetilde{g} \in \mathbb{G}_2$, $e(g^a, \widetilde{g}^b) = 1$ iff $a = 0$ or $b = 0$
- $e(\cdot, \cdot)$ is efficiently computable

In the sequel, the tuple $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p)$ is referred to as a bilinear environment. In addition, we will use $\{x_i\}_{i=a}^{i=b}$ to denote the set of elements x_i, $i \in [a, b]$ and $|\mathcal{B}|$ to denote the number of symbols that compose (i.e., the size) $\mathcal{B}$.

4.2 A Trivial Construction

A trivial attempt for defining a construction that ensures all of the security requirements we defined in Section 3.2 would consist of modifying the most efficient state of the art solution SEST towards a secret key based-construction as described in the following algorithms. The Setup, Keygen, and Encrypt algorithms are to be performed by the DO. The Issue algorithm will be performed collaboratively by the DO and the SE, while the Test algorithm will be performed by the SP.

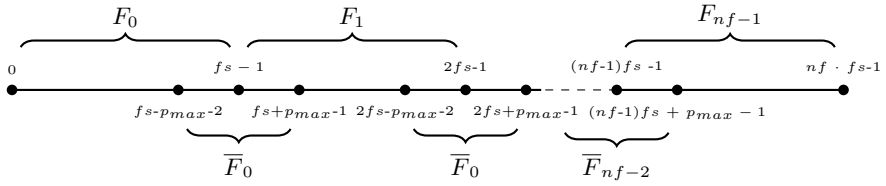

Fig. 1. Fragmentation approach

- **Setup**$(1^\lambda, n)$: Let $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_3, p)$ be a bilinear environment. This algorithm selects $g \xleftarrow{\$} \mathbb{G}_1, \widetilde{g} \xleftarrow{\$} \mathbb{G}_2$ and returns $params \leftarrow (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_3, p, g, \widetilde{g}, n)$.
- **Keygen**(Σ): On input of the alphabet Σ, this algorithm selects $z \xleftarrow{\$} \mathbb{Z}_p$ and $\{\alpha_\sigma \xleftarrow{\$} \mathbb{Z}_p\}_{\sigma \in \Sigma}$, computes and adds $\{g^{z^i}\}_{i=0}^{i=n-1}$ to $params$ (required for proving the trace indistinguishability property). It returns the secret key $K = \{z, \{\alpha_\sigma\}_{\sigma \in \Sigma}\}$.

- **Encrypt**$(\mathcal{B}, K)$: To encrypt $\mathcal{B} = \sigma_1 \cdots \sigma_n$, this algorithm chooses $a \xleftarrow{\$} \mathbb{Z}_p$ and returns $C = \{C_i, C'_i\}_{i=0}^{n-1}$ where $C_i = g^{a \cdot z^i}$ and $C'_i = g^{a \cdot \alpha_{\sigma_i} \cdot z^i}$.
- **Issue**(w, K) issues a trapdoor td_w for a pattern $w = \sigma_{w,0}, \cdots, \sigma_{w,l-1}$ of length $l \leq n$ as described in Algorithm 1.

Input: $K, params, w = \sigma_{w,0}, \cdots \sigma_{w,l-1}$
Output: td_w
$td_w = \emptyset$, $V = 0, c = 0$
$L[i] = 0$ for all $i \in [0, l-1]$
$Ind[\sigma] = 0$ for all $\sigma \in \Sigma$
foreach $i \in [0, l-1]$ **do**
  if $L[Ind[\sigma_{w,i}]] = 0$ **then**
    $L[c] \xleftarrow{\$} \mathbb{Z}_p$, $\mathcal{I}_c = \{i\}$, $c = c + 1$
  else
    $\mathcal{I}_{Ind[\sigma_{w,i}]} = \mathcal{I}_{Ind[\sigma_{w,i}]} \cup \{i\}$
  end
  $V = V + z^i \cdot \sigma_{w,i} \cdot L[Ind[\sigma_{w,i}]]$
  $Ind[\sigma_{w,i}] = Ind[\sigma_{w,i}] + 1$
end
$td_w = \{c, \{\mathcal{I}_j\}_{j=0}^{j=c-1}, \{\widetilde{g}^{L[j]}\}_{j=0}^{j=c-1}, \widetilde{g}^V\}$

Algorithm 1: Issue

- **Test**(C, td_w) tests whether the encrypted traces C contains w by parsing td_w as $\{c, \{\mathcal{I}_j\}_{j=0}^{j=c-1}, \{\widetilde{g}^{L[j]}\}_{j=0}^{j=c-1}, \widetilde{g}^V\}$ and C as $\{C_i, C'_i\}_{i=0}^{n-1}$, and checking for all $j \in [0, n-l]$ if the following equation holds:

$$\prod_{t=0}^{c-1} e(\prod_{i \in \mathcal{I}_t} C'_{j+i}, \widetilde{g}^{L[t]}) = e(C_j, \widetilde{g}^V)$$

We can prove the correctness, the trace indistinguishability, and encrypted trace unforgeability properties by following the same strategies as in Appendices A.1, A.2, and A.3. Unfortunately, this construction suffers mainly from three limitations. First, the size of the public parameters *params* is linear to the size of the data to be analyzed (which may be very large). Second, the pattern indistinguishability requirement cannot be satisfied since the Issue algorithm (Algorithm 1) leaks many information (such as, the number of different symbols and the maximum number of occurrences of a symbol) about the pattern to be searched. Third, searching the presence of a pattern w is linear to the maximum number of occurrences of each symbol in w, which makes this construction impractical for matching small alphabet based patterns (e.g., bit patterns).

4.3 P^{3}MED Construction

Overview The intuition behind our construction relies mainly on two observations. First, the number of analysis patterns is often very small compared to the

quantity of traces that are going to be analyzed (e.g., the number of patterns provided by SNORT is 3734 [22]). Second, the sizes of the analysis patterns are often very small compared to the size of the traces to be analyzed (e.g., the largest pattern size used by Snort is 364 Bytes).

For a trace with alphabet Σ, our construction associates each element σ of Σ with a secret encoding $(\alpha'_\sigma, \alpha_\sigma)$. It fragments the sequence of symbols that represents the traces $\mathcal{B}$ to be analyzed as described in Fig. 1 in which fs represents the number of symbols (i.e., the size) of each fragment and p_{max} represents the largest number of symbols in an analysis pattern. We require fs to be greater than or equals to $2 \cdot (p_{max} - 1)$ to ensure the correctness of the pattern matching functionality provided by our construction.

As illustrated in Fig. 1, the sequence of symbols $\mathcal{B}$ is fragmented into $2 \times nf - 1$ fragments $\{F_i, \overline{F}_j\}_{i=0,j=0}^{i=nf-1,j=nf-2}$ where $nf = |\mathcal{B}|/fs$ (for simplicity we will suppose that $|\mathcal{B}|$ is a multiple of fs). Each $F_i, i \in [0, nf-1]$ contains the symbols at indexes $[i \cdot fs, (i+1) \cdot fs - 1]$, while $\overline{F}_i, i \in [0, nf-2]$ contains the symbols at indexes $[(i+1) \cdot fs - p_{max} - 1, (i+1) \cdot fs + p_{max} - 1]$ of $\mathcal{B}$.

Given an $i \in [0, |\mathcal{B}| - 1]$, in the rest of this paper, we will denote by i_F the index of i inside the fragment F where $F \in \{F_0, \cdots, F_{nf-1}, \overline{F}_0, \cdots, \overline{F}_{nf-2}\}$. If $i \notin F$, i_F is not defined. Formally, assuming that $F = [a, b]$:

$$i_F = \begin{cases} i \bmod a \text{ if } i \in F \\ \emptyset \qquad\quad \text{otherwise} \end{cases}$$

A trapdoor for a pattern $w = \sigma_{w,0} \cdots \sigma_{w,l-1}$ will be associated with a set of polynomials $\{V_i = v_i \cdot \sum_{k=0}^{l-1} \alpha'_{\sigma_{w,k}} \cdot \alpha_{\sigma_{w,k}}^{k+i} \cdot z^k\}_{i=0}^{i=fs-1}$ where v_i is a random secret scalar used to prevent new trapdoor forgeries and z a random scalar belonging to the secret key K. The trapdoor of w consists then in the elements $\{\widetilde{g}^{V_i}, \widetilde{g}^{v_i}\}_{i=0}^{i=fs-1}$. Each couple $(\widetilde{g}^{V_i}, \widetilde{g}^{v_i})$ of the previous set will be used to check the presence of w at the index i of the previously constructed fragments.

Meanwhile, the encryption of each symbol σ_i of $\mathcal{B}$ is the tuple $\mathcal{C}_i = \{C_i, C'_i, \overline{C}_i, \overline{C'}_i\}$ that depends on the fragment in which its index i in $\mathcal{B}$ belongs. If the index i of σ_i belongs to F_ϵ (resp. $\overline{F}_\epsilon$) then C_i and C'_i (resp. $\overline{C}_i$ and $\overline{C'}_i$) contain the encryption of σ_i regarding the index i_{F_ϵ} of i in F_ϵ (resp. the index $i_{\overline{F}_\epsilon}$ of i in $\overline{F}_\epsilon$).

Then, if we want to test the presence of w at the index i, if i belongs to F_ϵ (resp. $\overline{F}_\epsilon$), then we compare the bilinear mapping results of the elements $C_{i_{F_\epsilon}}$, $\widetilde{g}^{v_{i_{F_\epsilon}}}$ (resp. $\overline{C}_{i_{\overline{F}_\epsilon}}$, $\widetilde{g}^{v_{i_{\overline{F}_\epsilon}}}$) and $C'_{i_{F_\epsilon}}$, $\widetilde{g}^{V_{i_{F_\epsilon}}}$ (resp. $\overline{C'}_{i_{\overline{F}_\epsilon}}$, $\widetilde{g}^{V_{i_{\overline{F}_\epsilon}}}$). If w is not present, then the results of the bilinear mapping will be random-looking polynomials which will be useless to the adversary.

We note that the choice of the fragmentation size fs as well as the alphabet Σ to be considered will determine the domain of the analysis patterns that can be queried. That is, the bigger the value of fs is, the bigger the sizes of the supported analysis patterns could be (since $fs \geq 2 \times p_{max}$). Moreover, the choice of the alphabet Σ will decide the "type" of search that can be performed by our construction. For instance, an hexadecimal alphabet where each symbol is represented in 4 bits will allow to perform hexadecimal searches over the

encrypted data, while an ASCII alphabet where each symbol is represented in 8 bits will allow to perform ASCII searches over the encrypted data. In Section 7, we empirically analyze the effects of the variation of the values of fs and Σ on the practicality of our construction.

In the remaining of this paper, for simplicity, we will suppose that $|\mathcal{B}|$ is a multiple of fs.

The Protocol

- **Setup**$(1^\lambda, p_{max})$: Let $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_3, p)$ be a bilinear environment. This algorithm selects $g \xleftarrow{\$} \mathbb{G}_1, \widetilde{g} \xleftarrow{\$} \mathbb{G}_2$ and returns $params \leftarrow (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_3, p, g, \widetilde{g}, p_{max})$.
- **Keygen**(Σ): On input of the alphabet Σ, this algorithm chooses fs such that $fs \geq 2 \cdot (p_{max} - 1)$, selects $z \xleftarrow{\$} \mathbb{Z}_p$ and $\{\alpha'_\sigma, \alpha_\sigma \xleftarrow{\$} \mathbb{Z}_p\}_{\sigma \in \Sigma}$, computes and adds $pp \leftarrow \{g^{z^i}\}_{i=0}^{i=fs-1}$ to $params$. It returns the secret key $K = \{z, \{\alpha'_\sigma, \alpha_\sigma\}_{\sigma \in \Sigma}\}$.
- **Encrypt**$(\mathcal{B}, K)$: This algorithm starts by fragmenting $\mathcal{B} = \sigma_0, \cdots \sigma_{m-1}$ into $\{F_i, \overline{F}_j\}_{i=0,j=0}^{i=nf-1,j=nf-2}$ where F_i (resp. $\overline{F}_j$) contains the symbols of $\mathcal{B}$ at indices $[i \cdot fs, (i+1) \cdot fs - 1]$ (resp. $[(j+1) \cdot fs - p_{max} - 2, (j+1) \cdot fs + p_{max} - 1]$). It chooses $a_k \xleftarrow{\$} \mathbb{Z}_p$ for each $k \in [0, nf-1]$ and $\overline{a}_k \xleftarrow{\$} \mathbb{Z}_p$ for each $k \in [0, nf-2]$ and returns $C = \{C_i, \overline{C}_i, C'_i, \overline{C'}_i\}_{i=0}^{m-1}$ as described in Algorithm 2.

Input: $K, params, \mathcal{B} = \sigma_0, \cdots \sigma_{m-1}, \{F_i, a_i, \overline{F}_j, \overline{a}_j\}_{i=0,j=0}^{i=nf-1,j=nf-2}$
Output: $C = \{C_i, \overline{C}_i, C'_i, \overline{C'}_i\}_{i=0}^{m-1}$
$C \leftarrow \emptyset$
foreach $i \in [0, m-1]$ **do**
  $\epsilon \leftarrow i/fs$ #find the fragment F_ϵ to which i belongs
  $C_i \leftarrow g^{a_\epsilon \cdot {\alpha'_{\sigma_i}}^{\cdot (\alpha_{\sigma_i} \cdot z)^{i_{F_\epsilon}}}}$, $C'_i \leftarrow g^{a_\epsilon \cdot z^{i_{F_\epsilon}}}$
  if $\epsilon > 0$ ***and*** $i \in \overline{F}_{\epsilon-1}$ **then**
    $\overline{C}_i \leftarrow g^{\overline{a}_{\epsilon-1} \cdot {\alpha'_{\sigma_i}}^{\cdot (\alpha_{\sigma_i} \cdot z)^{i_{\overline{F}_{\epsilon-1}}}}}$
    $C'_i \leftarrow g^{\overline{a}_{\epsilon-1} \cdot z^{i_{\overline{F}_{\epsilon-1}}}}$
  else if $\epsilon < nf - 1$ ***and*** $i \in \overline{F}_\epsilon$ **then**
    $\overline{C}_i \leftarrow g^{\overline{a}_\epsilon \cdot {\alpha'_{\sigma_i}}^{\cdot (\alpha_{\sigma_i} \cdot z)^{i_{\overline{F}_\epsilon}}}}$, $C'_i \leftarrow g^{\overline{a}_\epsilon \cdot z^{i_{\overline{F}_\epsilon}}}$
  else
    $\overline{C}_i \leftarrow$ Null, $\overline{C'}_i \leftarrow$ Null
  end
  $C \leftarrow C \cup \{C_i, C'_i, \overline{C}_i, \overline{C'}_i\}$
end

Algorithm 2: Encrypt

- **Issue**(w, K) issues a trapdoor td_w for the sequence of symbols $w = \sigma_{w,0}, \cdots, \sigma_{w,l-1}$ of Σ of length $l < p_{max}$ as described in Algorithm 3.

Input: $K, params, w = \sigma_{w,0}, \cdots \sigma_{w,l-1}$
Output: $td_w = \{V_i, v_i\}_{i=0}^{fs-l}$
$td_w \leftarrow \emptyset$
foreach $i \in [0, fs-l]$ **do**
  $v_i \xleftarrow{\$} \mathbb{Z}_p$
  $V_i = v_i \cdot \sum_{k=0}^{l-1} \alpha'_{\sigma_{w,k}} \cdot \alpha^{k+i}_{\sigma_{w,k}} \cdot z^k$
  $td_w \leftarrow td_w \cup \{\widetilde{g}^{V_i}, \widetilde{g}^{v_i}\}$
end

Algorithm 3: Issue

– **Test**(C, td_w) tests whether the encrypted traces C contains w using Algorithm 4. It returns the set $\mathcal{I}$ of indexes i in which w exists in C.

Input: $C = \{C_i, \overline{C}_i, C'_i, \overline{C'}_i\}_{i=0}^{m-1}, td_w = \{V_i, v_i\}_{i=0}^{i=fs-l}$
Output: $\mathcal{I}$
$\mathcal{I} \leftarrow \emptyset$
foreach $i \in [0, m-l]$ **do**
  $\epsilon \leftarrow i/fs$ #find the fragment F_ϵ to which the index i belongs
  if $\epsilon < nf - 1$ ***and*** $i \in F_\epsilon \cap \overline{F}_\epsilon$ **then**
    if $e(\prod_{j=0}^{l-1} \overline{C}_{i+j}, \widetilde{g}^{v_{i_{\overline{F}_\epsilon}}}) = e(\overline{C'}_i, \widetilde{g}^{V_{i_{\overline{F}_\epsilon}}})$ **then**
      $\mathcal{I} \leftarrow \mathcal{I} \cup i$
    end
  else
    if $e(\prod_{j=0}^{l-1} C_{i+j}, \widetilde{g}^{v_{i_{F_\epsilon}}}) = e(C'_i, \widetilde{g}^{V_{i_{F_\epsilon}}})$ **then**
      $\mathcal{I} \leftarrow \mathcal{I} \cup i$
    end
  end
end

Algorithm 4: Test

We note here that the size of the ciphertext produced by the Encrypt algorithm does not depend on either the sizes or the number of the analysis patterns to be used but depends only on the size of data to be encrypted. In addition, our Issue and Test algorithms allow to search an arbitrary (upper bounded size) and unforgeable (without the knowledge of the secret key K) patterns. Moreover, the sizes of the issued trapdoors do not depend on the size of the data to be encrypted but only on the size of the data fragment (around the double of the maximum size of an analysis pattern).

5 Security

As in [1], the security of our construction holds as long as $\mathbb{G}_1 \neq \mathbb{G}_2$ and no efficiently computable homomorphism exists between $\mathbb{G}_1$ and $\mathbb{G}_2$ in either directions. We prove the security of our construction under an interactive assumption. That is, we use a slightly modified General Diffie-Hellman (GDH) problem assumption [21] to allow the adversary to request the set of values on which the reduction will break the GDH assumption. This interactive aspect of the GDH instance we are considering reduces slightly the security of the construction we are proposing. However, this interactive assumption allowed as to define a quite efficient construction with interesting properties. First, the size of the ciphertext depends only on the size of the plaintext (it is independent of the sizes of the analysis patterns). Second, the size of the issued trapdoors is independent of the size of the trace to be analyzed. Third, the search complexity depends only on the size of the trace and is constant on the size of the analysis patterns. Attaining all previously mentioned properties while being able to handle arbitrary (upper bounded size) analysis pattern query is not obvious and may justify the use of such an interactive assumption.

5.1 Security Assumption

Following the security model described in Section 3.4, We prove the trace and detection pattern indistinguishability under the i-GDH assumption introduced in [1]. The complexity of the i-GDH assumption depends mainly on the *independance* property we formalize in the following Definition.

Definition 5 (independence). *Let p be some large prime, r, s, t, c, and k be five positive integers and $R \in \mathbb{F}_p[X_1, \cdots, X_c]^r$, $S \in \mathbb{F}_p[X_1, \cdots, X_c]^s$, and $T \in \mathbb{F}_p[X_1, \cdots, X_c]^t$ be three tuples of multivariate polynomials over $\mathbb{F}_p$. Let $R^{(i)}$, $S^{(i)}$ and $T^{(i)}$ denote respectively the i-th polynomial contained in R, S, and T. For any polynomial $f \in \mathbb{F}_p[X_1, \cdots, X_c]$, we say that f is dependent on $< R, S, T >$ if there exist constants $\{\vartheta_j^{(a)}\}_{j=1}^{s}$, $\{\vartheta_{i,j}^{(b)}\}_{i=1,j=1}^{i=r,j=s}$, $\{\vartheta_k^{(c)}\}_{k=1}^{t}$ such that*

$$f \cdot (\sum_j \vartheta_j^{(a)} \cdot S^{(j)}) = \sum_{i,j} \vartheta_{i,j}^{(b)} \cdot R^{(i)} \cdot S^{(j)} + \sum_k \vartheta_k^{(c)} T^{(k)}$$

We say that f is independent of $< R, S, T >$ if f is not dependent on $< R, S, T >$.

Definition 6 (i-GDH assumption). *Let p be some large prime, r, s, t, c, and k be five positive integers and $R \in \mathbb{F}_p[X_1, \cdots, X_c]^r$, $S \in \mathbb{F}_p[X_1, \cdots, X_c]^s$, and $T \in \mathbb{F}_p[X_1, \cdots, X_c]^t$ be three tuples of multivariate polynomials over $\mathbb{F}_p$. Let $\mathcal{O}^R$, (resp. $\mathcal{O}^S$ and $\mathcal{O}^T$) be oracle that, on input $\{\{a_{i_1,\cdots,i_c}^{(k)}\}_{i_j=0}^{d_k}\}_k$, adds the polynomials $\{\sum_{i_1,\cdot,i_c} a_{i_1,\cdot,i_c}^{(k)} \prod_j X_j^{i_j}\}_k$ to R (resp. S and T).*

Let $(x_1, \cdots, x_c)$ be secret vector and q_r (resp. q_s) (resp. q_t) be the number of queries to $\mathcal{O}^R$ (resp. $\mathcal{O}^S$) (resp. $\mathcal{O}^T$). The i-GDH assumption states that, given the values $\{g^{R^{(i)}(x_1,\cdots,x_c)}\}_{i=1}^{r+k\cdot q_r}$, $\{\widetilde{g}^{S^{(i)}(x_1,\cdots,x_c)}\}_{i=1}^{s+k\cdot q_s}$, and $\{e(g,\widetilde{g})^{T^{(i)}(x_1,\cdots,x_c)}\}_{i=1}^{t+k\cdot q_t}$,

it is hard to decide whether $U = g^{f(x_1,\cdots,x_c)}$ *or* U *is random if* f *is independent of* $< R, S, T >$.

As argued in [1], The hardness of the i-GDH problem depends on the same argument as the GDH problem which has already been proven in the generic group model [21]. That is, as long as the challenge polynomial that we denote f is independent of $< R, S, T >$, an adversary cannot distinguish $g^{f(x_1,\cdots,x_c)}$ from a random element of $\mathbb{G}_1$. The definition method of the content of the sets R,S, and T (by assumption or by the queries to oracles) does not fundamentally change the proof.

5.2 Security Results

In this section, we prove that the P^{3}MED construction described in Section 4.3 provides the security requirements we described in Section 3.2. Proofs of Theorems 2, 3, and 4 are given in Appendix A.

Theorem 1. *P^{3}MED is correct.*

Theorem 2. *P^{3}MED is trace indistinguishable under the i-GDH assumption.*

Theorem 3. *P^{3}MED is encrypted trace forgery secure under the i-GDH assumption.*

Theorem 4. *P^{3}MED is pattern indistinguishable under the i-GDH assumption.*

6 The complexity

We evaluate the practicability of P^{3}MED regarding several properties: the sizes of the public parameters, the ciphertext, the trapdoor, and the encryption and search complexities. Let fs be the size of a fragment, p_{max} be the maximum size of a pattern, n be the total number of symbols in the data to be analyzed.

The size of the public parameters: The public parameters used in our construction contains fs elements of $\mathbb{G}_1$ which represents $32 \times fs$ bytes using Barreto-Naehrig (BN) [16].

The size of the ciphertext: In the worst case (i.e., $fs = 2 \times (p_{max} - 1)$), each symbol will be represented by 4 elements of $\mathbb{G}_1$. Therefore, encrypting n symbols requires $128 \times n$ bytes using BN.

Trapdoor's size: A trapdoor is composed of $2 \times (fs - p_{max})$ elements of $\mathbb{G}_2$ which represents $64 \times (fs - p_{max})$ bytes using BN.

Trapdoor generation complexity. Generating a trapdoor for a pattern of length l ($l \leq p_{max}$), as described in the Issue algorithm (Algorithm 3), requires $(fs - l) \times (2l + 2)$ exponentiations and $4l(fs - l)$ multiplications in $\mathbb{G}_2$.

Encryption complexity According to the Encrypt algorithm (Algorithm 2),

In the worst case (i.e., $fs = 2 \times (p_{max} - 1)$), encrypting a sequence of n symbols requires $10 \times n$ exponentiations in $\mathbb{G}_1$. In case in which the size of the data to encrypt is large (i.e., the fragment size fs and the size of the considered alphabet Σ is negligible compared to the size of the data to be encrypted), the previous complexity can be reduced by pre-computing $\{g^{\alpha'_\sigma \cdot (\alpha_\sigma \times z)^i}, g^{z^i}\}_{i=0,\sigma\in\Sigma}^{i=fs-1}$. Then for each symbol to encrypt, the encryptor needs only to perform two exponentiations: $(g^{\alpha'_\sigma \cdot (\alpha_\sigma \times z)^i})^{a_j}$ and $(g^{z^i})^{a_j}$ which reduces the overall complexity to $fs \times |\Sigma| + 2 \times n$ exponentiations in $\mathbb{G}_1$.

Input: $C = \{C_i, \overline{C}_i, C'_i, \overline{C'}_i\}_{i=0}^{m-1}, td_w = \{V_i, v_i\}_{i=0}^{i=fs-l}$
Output: $\mathcal{I}$
$\mathcal{I} \leftarrow \emptyset$
$C_E = 0$
foreach $i \in [0, m-l]$ **do**
  $\epsilon \leftarrow i/fs$ #find the fragment F_ϵ to which i belongs
  if $\epsilon < nf - 1$ ***and*** $i \in F_\epsilon \cap \overline{F}_\epsilon$ **then**
    if $i = 0$ **then**
      $C_E = \prod_{j=0}^{l-1} \overline{C}_{i+j}$
    else
      $C_E = (\frac{C_E}{\overline{C}_{i-1}}) * \overline{C}_{i+j}$
    end
    if $e(C_E, \widetilde{g}^{v_{i_{\overline{F}_\epsilon}}}) = e(\overline{C'}_i, \widetilde{g}^{V_{i_{\overline{F}_\epsilon}}})$ **then**
      $\mathcal{I} \leftarrow \mathcal{I} \cup i$
    end
  else
    if $i = 0$ **then**
      $C_E = \prod_{j=0}^{l-1} C_{i+j}$
    else
      $C_E = (\frac{C_E}{C_{i-1}}) * C_{i+j}$
    end
    if $e(\prod_{j=0}^{l-1} C_{i+j}, \widetilde{g}^{v_{i_{F_\epsilon}}}) = e(C'_i, \widetilde{g}^{V_{i_{F_\epsilon}}})$ **then**
      $\mathcal{I} \leftarrow \mathcal{I} \cup i$
    end
  end
end

Algorithm 5: Optimized Test Algorithm (Optimization instructions are highlighted)

Search complexity: According to the Test algorithm (Algorithm 4), searching a pattern of size l on a sequence of symbols of size n requires $nl - l^2$ multiplications on the group $\mathbb{G}_1$ and $2 \times (n - l)$ pairings. In fact, the Test algorithm verifies the presence of a pattern (using its associated trapdoor) in each possible offset

in the sequence of symbols to be analyzed. Let us denote by Σ_0 and Σ_1 the two sequences of symbols to be analyzed to check the presence of a pattern in offsets 0 and 1 respectively of the fragment F_i (resp. $\overline{F}_i$). Checking the presence of the pattern in the offset 0 requires the computation of $\prod_{i=0}^{l-1} C_i$ (resp. $\prod_{i=0}^{l-1} \overline{C}_i$) while checking the presence of the pattern in offset 1 requires the computation of $\prod_{i=0}^{l-1} C_{i+1}$ (resp. $\prod_{i=0}^{l-1} \overline{C}_{i+1}$). Obviously, for the offset 1, we can avoid the recomputation of $\prod_{i=1}^{l-1} C_i$ since it has already been computed for the offset 0. Following the previous observation, we optimize the Test algorithm as illustrated in Algorithm 5. Consequently, by using the new optimized algorithm, searching a pattern of length l on a sequence of symbols of length n requires only n multiplications and n divisions on the group $\mathbb{G}_1$, and $2\times(n-l)$ pairings. Considering the fact that the size l of pattern to be searched is pretty often negligible compared to the size n of the sequence of symbols to be analyzed, we can upper bound the search complexity by n multiplications, n divisions and $2n$ pairings.We note that pairing operations can be implemented very efficiently [17] and that our Test procedure (Algorithm 5) is highly parallelizable.

7 Empirical Evaluation

In this section, we experimentally evaluate the performance of P^{3}MED. We implement our construction using the optimal Ate library [17], over the 254-bits Barreto-Naehrig curve. It consists of 1196 lines of code excluding 96 lines for testing purposes. For all conducted experiments, we used real network traces as the data to be encrypted and analyzed, and we (pseudo) randomly generated the analysis patterns to be searched. In addition, since the encryption and the trapdoor generation algorithms are to be performed by data owners which may not have a large computation power, we run both the trapdoor generation and the encryption algorithms tests on an Amazon EC2 instance (a1.2xlarge) running Linux with an Intel Xeon E5-2680 v4 Processor with 8 vCPU and 16 GB of RAM. In contrast, as the search operations (Algorithm 5) are performed by the SP which is supposed to have a large computation power, we run our experiments on an Amazon EC2 instance (m5.24xlarge) running Linux with an Intel Xeon E5-2680 v4 Processor with 96 vCPU and 64 GB of RAM.

In our empirical evaluation, we aim to quantify the following characteristics of our construction:

- The time required to generate a trapdoor and its corresponding size as a function of the size of the largest analysis pattern p_{max} that can be searched.
- The time taken to encrypt a trace as a function of its size (i.e. the size of the sequence of symbols that composed the data to be encrypted), the fragmentation size fs and the size of the considered alphabet.
- The time needed to perform a pattern matching query (Algorithm 5) as a function of the size of the data to be queried and the size of the analysis pattern to be searched.

Trapdoor generation. Fig. 2 describes the time required for issuing a trapdoor for an analysis pattern w as a function of its length (i.e., the number of symbols in w). According to our experiments, issuing a trapdoor for a pattern of 4000 symbols take less than a second. In addition, the sizes of the generated trapdoors are relatively small (256 KB for a pattern of 4000 symbols).

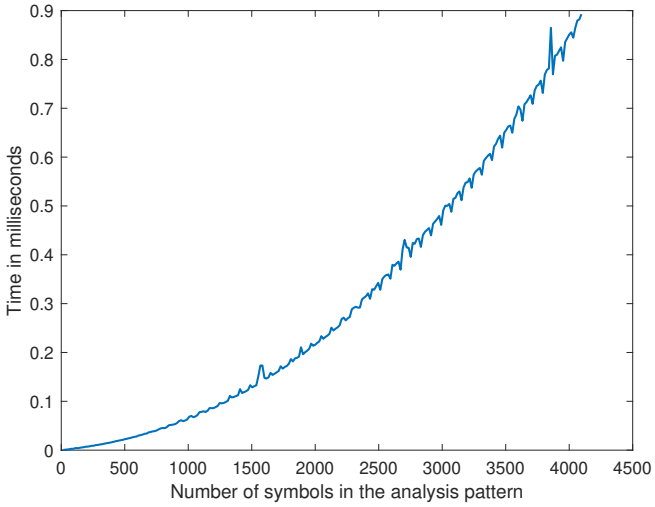

Fig. 2. Encryption time as a function of the number of symbols in the data to be encrypted

Encryption time. According to Section 6, the duration of an encryption operation depends mainly on the number of symbols in the data to be encrypted n but also on the fragmentation size fs and the size $|\Sigma|$ of the considered alphabet Σ. Fig. 3 describes the time needed to encrypt a network trace $\mathcal{T}$ fragmented in chunks, each containing 1000 bits ($fs = 1000$ and $\Sigma = \{0, 1\}$), as a function of n. The encryption of 1 Gb of data takes around 77 seconds with a multi-threaded implementation [3], which shows that the encryption algorithm used by our construction is quite efficient even for large datasets.

As we noted in Section 4.3, the fragmentation size fs and the considered alphabets are important parameters in our construction. The first directly influences the size of the largest analysis pattern that can be searched over the encrypted data since the bigger the size of the fragments are, the bigger the size of the supported analysis patterns could be. The second parameter determines the type of search that can be performed by our construction. In Fig. 4, we compute the time required for the encryption of a dataset composed of 10^8 symbols

[3] Encryption time would be roughly 8 times slower with a single-threaded execution.

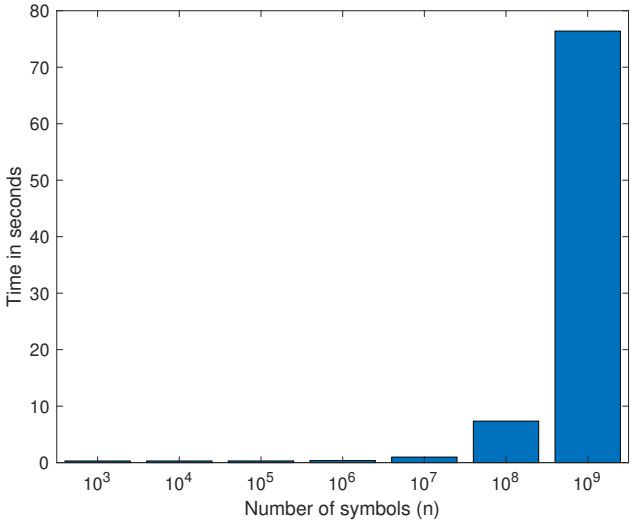

Fig. 3. Encryption time as a function of the number of symbols in the data to be encrypted

as a function of the fragmentation size fs and the type of the considered symbols. We consider three types of alphabets: binary, hexadecimal, and base 256 (i.e., ASCII alphabet) where each symbol is represented respectively in 1, 4 and 8 bits. For fs, we consider 3 different fragment sizes: 10^3, 10^4, and 10^5 symbols.

As illustrated in Fig. 4, the time required for encrypting a dataset composed of 10^8 symbols increases only by a factor of 0.02 (from 7,04 to 7,2 seconds) when increasing the size of the fragments by a factor of 100 (from 10^3 to 10^5) and increasing the size of the considered alphabet by a factor of 128 (from a base 2 alphabet where $\Sigma = \{0, 1\}$ to a base 256 alphabet where $\Sigma = \{0, 1, \cdots, 255\}$).

Search time. As shown in Section 6, the complexity of the search operation (Algorithm 5) depends mainly on the number of encrypted symbols n that compose the data to be analyzed. Fig. 5 describes the time required for searching a pattern as a function of the number of encrypted symbols in the data to be analyzed.

The conducted evaluations show that the average search throughput of our construction is 139078 symbols per second with a multi-threaded implementation [4]. Thus, if an ASCII (resp. binary) alphabet is considered, the search throughput is 139 KB (resp. Kb) per second. This remains a quite efficient throughput for pattern matching over encrypted data compared to what can be performed by the most efficient state of the art solution SEST which offers similar features (i.e., correct analysis as well as trace indistinguishably). Fig. 6 (resp. Fig. 7)

[4] search time would be roughly 100 times slower with a single-threaded execution.

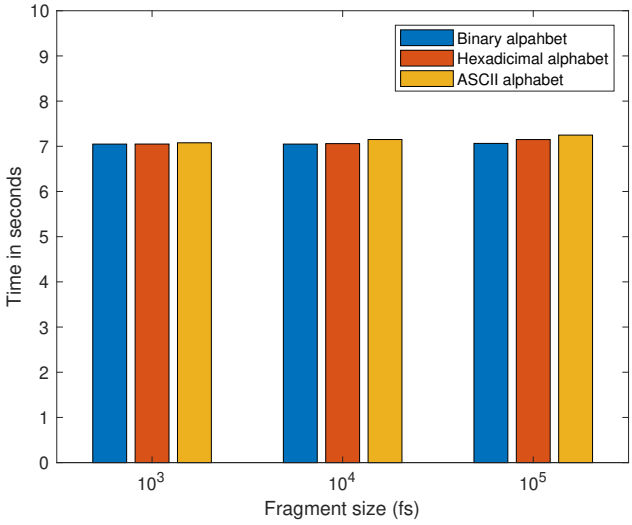

Fig. 4. Time required for encrypting 10^8 symbols as a function of the fragmentation size fs and the type of the considered alphabet

compares the time needed for both our and the SEST (both its asymmetric [1] and symmetric (Section 4.2) variants) constructions to test the presence of a pattern of bits (resp. of bytes) in a 10 MB (resp. Mb) dataset as a function of the length of the pattern to be searched. In both bit and byte searches, our construction drastically reduces the search time compared to SEST. This is mainly due to the fact that our Test algorithm (Algorithm 5) is constant on the size and on the content of the searched pattern which is not the case for SEST.

8 Conclusion

In this work, we introduced a new provably correct, secure, and quite efficient construction that operates pattern matching directly over encrypted traffic. The proposed construction has several remarkable properties. First, it ensures data and pattern indistinguishability meaning that the entity that is going to perform pattern matching will learn nothing about the patterns to be searched as well as the data to be inspected, except the presence or the absence of a set of "unknown" patterns (since the entity charged to perform pattern matching will not have access to the patterns plaintexts). Second, the size of the ciphertext is linear to the size of the plaintext and is constant on the sizes and the number of analysis patterns. Third, the size of the issued trapdoors is constant on the size of the data to be analyzed. Finally, the search complexity is linear to the size of the trace and is constant on the size of the analysis patterns. The proposed

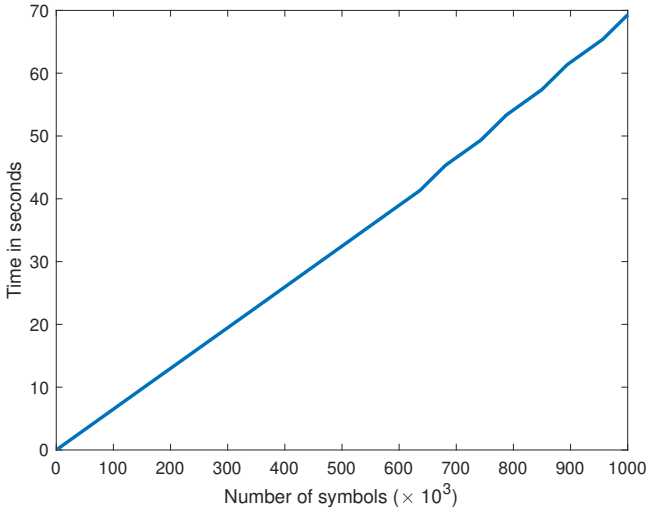

Fig. 5. Time required for searching a pattern as a function of the number of encrypted symbols in the data to be analyzed

construction can be useful for other application scenarios such as subtrees search and searching of structured data.

References

1. Desmoulins, N., Fouque, P. A., Onete, C., & Sanders, O. (2018, December). Pattern Matching on Encrypted Streams. In International Conference on the Theory and Application of Cryptology and Information Security (pp. 121-148). Springer, Cham.
2. Moataz, T., Justus, B., Ray, I., Cuppens-Boulahia, N., Cuppens, F., & Ray, I. (2014, July). Privacy-preserving multiple keyword search on outsourced data in the clouds. In IFIP Annual Conference on Data and Applications Security and Privacy (pp. 66-81). Springer, Berlin, Heidelberg.
3. Curtmola, R., Garay, J., Kamara, S., & Ostrovsky, R. (2011). Searchable symmetric encryption: improved definitions and efficient constructions. Journal of Computer Security, 19(5), 895-934.
4. Kamara, S., Moataz, T., & Ohrimenko, O. (2018, August). Structured encryption and leakage suppression. In Annual International Cryptology Conference (pp. 339-370). Springer, Cham.
5. Chase, M., & Shen, E. (2015). Substring-searchable symmetric encryption. Proceedings on Privacy Enhancing Technologies, 2015(2), 263-281.
6. Sherry, J., Lan, C., Popa, R. A., & Ratnasamy, S. (2015). Blindbox: Deep packet inspection over encrypted traffic. ACM SIGCOMM Computer communication review, 45(4), 213-226.
7. Canard, S., Diop, A., Kheir, N., Paindavoine, M., & Sabt, M. (2017, April). Blindids: Market-compliant and privacy-friendly intrusion detection system over encrypted

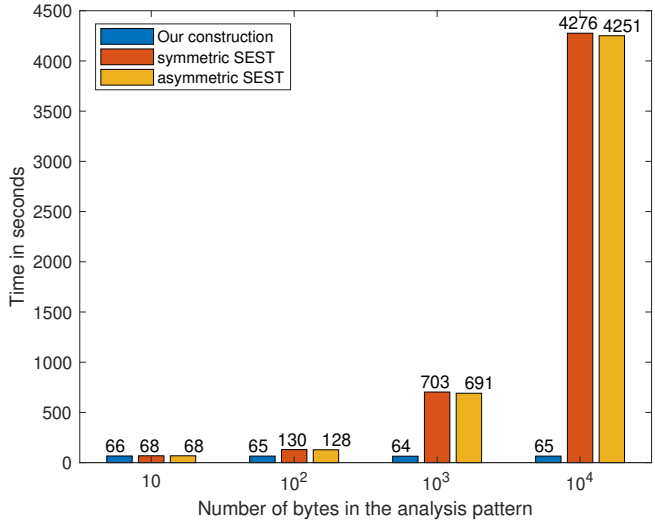

Fig. 6. Timing comparison for testing the presence of a pattern in a string of 10 MB as a function of the pattern size using the SEST schema and our construction.

traffic. In Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security (pp. 561-574). ACM.

8. Gentry, C., & Boneh, D. (2009). A fully homomorphic encryption scheme (Vol. 20, No. 09). Stanford: Stanford University.
9. Boneh, D., Sahai, A., & Waters, B. (2011, March). Functional encryption: Definitions and challenges. In Theory of Cryptography Conference (pp. 253-273). Springer, Berlin, Heidelberg.
10. Lauter, K., López-Alt, A., & Naehrig, M. (2014, September). Private computation on encrypted genomic data. In International Conference on Cryptology and Information Security in Latin America (pp. 3-27). Springer, Cham.
11. Hazay, C., & Lindell, Y. (2010). Efficient protocols for set intersection and pattern matching with security against malicious and covert adversaries. Journal of cryptology, 23(3), 422-456.
12. Gennaro, R., Hazay, C., & Sorensen, J. S. (2016). Automata evaluation and text search protocols with simulation-based security. Journal of Cryptology, 29(2), 243-282.
13. Troncoso-Pastoriza, J. R., Katzenbeisser, S., & Celik, M. (2007, October). Privacy preserving error resilient DNA searching through oblivious automata. In Proceedings of the 14th ACM conference on Computer and communications security (pp. 519-528). ACM.
14. Katz, J., Sahai, A., & Waters, B. (2013). Predicate encryption supporting disjunctions, polynomial equations, and inner products. Journal of cryptology, 26(2), 191-224.
15. Boneh, D., & Waters, B. (2007, February). Conjunctive, subset, and range queries on encrypted data. In Theory of Cryptography Conference (pp. 535-554). Springer, Berlin, Heidelberg.

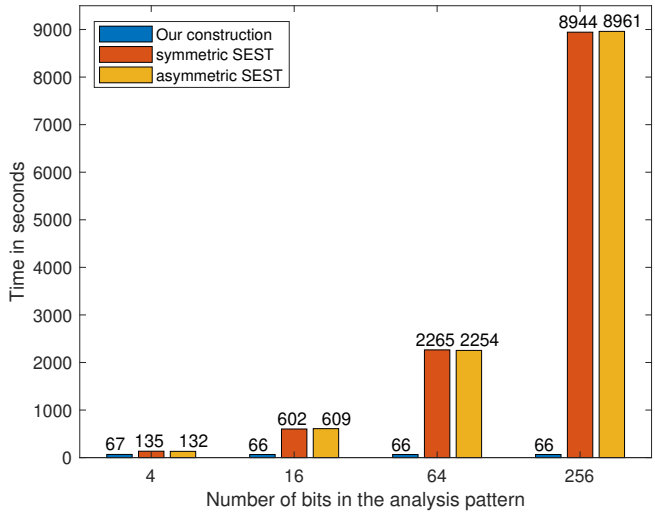

Fig. 7. Timing comparison for testing the presence of a pattern in a string of 10 MB as a function of the pattern size using the SEST schema and our construction.

16. Barreto, P. S., & Naehrig, M. (2005, August). Pairing-friendly elliptic curves of prime order. In International Workshop on Selected Areas in Cryptography (pp. 319-331). Springer, Berlin, Heidelberg.
17. Mitsunari, S. (2013). A Fast Implementation of the Optimal Ate Pairing over BN curve on Intel Haswell Processor. IACR Cryptology ePrint Archive, 2013, 362.
18. Canetti, R., Halevi, S., & Katz, J. (2003, May). A forward-secure public-key encryption scheme. In International Conference on the Theory and Applications of Cryptographic Techniques (pp. 255-271). Springer, Berlin, Heidelberg.
19. Bellare, M., Boldyreva, A., & O'Neill, A. (2007, August). Deterministic and efficiently searchable encryption. In Annual International Cryptology Conference (pp. 535-552). Springer, Berlin, Heidelberg.
20. MISP - Open Source Threat Intelligence Platform & Open Standards For Threat Information Sharing, https://www.misp-project.org/, 23 12 2011.
21. Boyen, X. (2008, September). The uber-assumption family. In International Conference on Pairing-Based Cryptography (pp. 39-56). Springer, Berlin, Heidelberg.
22. Snort Rules. https://www.snort.org/, Accessed: 2019-08-35.

A Appendix: Security Proofs

A.1 Proof of Theorem 1

Let us suppose that $\mathcal{B} = \sigma_0 \cdots \sigma_{m-1}$ contains $w = \sigma_{w,0} \cdots \sigma_{w,l-1} (l < p_{max}, l < m)$ at index i. Thus, $\forall j \in [0, l-1] : \sigma_{i+j} = \sigma_{w,j}$. Let us suppose that $i \in F_\epsilon (\epsilon \in [0, nf-1])$. According to the Test algorithm (Algorithm 4), 2 cases should be considered:

– **Case 1:** $\epsilon < nf - 1$ **and** $i \in F_\epsilon \cap \overline{F}_\epsilon$

$$\begin{aligned} e(\prod_{j=0}^{l-1} \overline{C}_{i+j}, \widetilde{g}^{v_{i_{\overline{F}_\epsilon}}}) &= e(\prod_{j=0}^{l-1} g^{\overline{a}_\epsilon \cdot \alpha'_{\sigma_{i+j}} \cdot (\alpha_{\sigma_{i+j}} \cdot z)^{(i_{\overline{F}_\epsilon}+j)}}, \widetilde{g}^{v_{i_{\overline{F}_\epsilon}}}) \\ &= e(g^{\overline{a}_\epsilon \cdot \sum_{j=0}^{l-1} \alpha'_{\sigma_{i+j}} \cdot (\alpha_{\sigma_{i+j}} \cdot z)^{(i_{\overline{F}_\epsilon}+j)}}, \widetilde{g}^{v_{i_{\overline{F}_\epsilon}}}) \\ &= e(g^{\overline{a}_\epsilon \cdot z^{i_{\overline{F}_\epsilon}}}, \widetilde{g}^{v_{i_{\overline{F}_\epsilon}} \cdot \sum_{j=0}^{l-1} \alpha'_{\sigma_{i+j}} \cdot \alpha_{\sigma_{i+j}}^{i_{\overline{F}_\epsilon}+j} \cdot z^j}) \end{aligned}$$

By replacing $\alpha'_{\sigma_{i+j}}$ by $\alpha'_{\sigma_{w,j}}$ and $\alpha_{\sigma_{i+j}}$ by $\alpha_{\sigma_{w,j}}$ we get:

$$e(\prod_{j=0}^{l-1} \overline{C}_{i+j}, \widetilde{g}^{v_{i_{\overline{F}_\epsilon}}}) = e(\overline{C'}_i, \widetilde{g}^{V_{i_{\overline{F}_\epsilon}}})$$

– **Case 2:** $i \in F_\epsilon \backslash \overline{F}_\epsilon$

$$\begin{aligned} e(\prod_{j=0}^{l-1} C_{i+j}, \widetilde{g}^{v_{i_{F_\epsilon}}}) &= e(\prod_{j=0}^{l-1} g^{a_\epsilon \cdot \alpha'_{\sigma_{i+j}} \cdot (\alpha_{\sigma_{i+j}} \cdot z)^{(i_{F_\epsilon}+j)}}, \widetilde{g}^{v_{i_{F_\epsilon}}}) \\ &= e(g^{a_\epsilon \cdot \sum_{j=0}^{l-1} \alpha'_{\sigma_{i+j}} \cdot (\alpha_{\sigma_{i+j}} \cdot z)^{(i_{F_\epsilon}+j)}}, \widetilde{g}^{v_{i_{F_\epsilon}}}) \\ &= e(g^{a_\epsilon \cdot z^{i_{F_\epsilon}}}, \widetilde{g}^{v_{i_{F_\epsilon}} \cdot \sum_{j=0}^{l-1} \alpha'_{\sigma_{i+j}} \cdot \alpha_{\sigma_{i+j}}^{i_{F_\epsilon}+j} \cdot z^j}) \end{aligned}$$

Again, by replacing $\alpha'_{\sigma_{i+j}}$ by $\alpha'_{\sigma_{w,j}}$ and $\alpha_{\sigma_{i+j}}$ by $\alpha_{\sigma_{w,j}}$ we get:

$$e(\prod_{j=0}^{l-1} C_{i+j}, \widetilde{g}^{v_{i_{F_\epsilon}}}) = e(C'_i, \widetilde{g}^{V_{i_{F_\epsilon}}})$$

As a result in both previous cases, the probability that the **Test** algorithm returns a set containing i is 1.

For the second part of the proof, we assume that the set of indexes returned by **Test** contains i despite that $\sigma_i \cdots \sigma_{i+l-1} \neq \sigma_{w,i} \cdots \sigma_{w,l-1}$. We should consider the following two cases:

– **Case 1:** $\epsilon < nf - 1$ **and** $i \in F_\epsilon \cap \overline{F}_\epsilon$. Let us denote by $\mathcal{K}_{\neq}$ the non-empty set of indexes k in which $\sigma_{i+k} \neq \sigma_{w,k}$ ($k \in [0, l-1]$). Since i has been returned by **Test**, then we have:

$$\begin{aligned} & e(\prod_{j=0}^{l-1} \overline{C}_{i+j}, \widetilde{g}^{v_{i_{\overline{F}_\epsilon}}}) = e(\overline{C'}_i, \widetilde{g}^{V_{i_{\overline{F}_\epsilon}}}) \\ \Leftrightarrow \quad & e(g^{\overline{a}_\epsilon \cdot \sum_{k=0}^{l-1} \alpha'_{\sigma_{i+k}} \cdot (\alpha_{\sigma_{i+k}} \cdot z)^{(i_{\overline{F}_\epsilon}+k)}}, \widetilde{g}^{v_{i_{\overline{F}_\epsilon}}}) \\ & \qquad = e(g^{\overline{a}_\epsilon \cdot z^{i_{\overline{F}_\epsilon}}}, \widetilde{g}^{v_{i_{\overline{F}_\epsilon}} \cdot \sum_{k=0}^{l-1} \alpha'_{\sigma_{i+k}} \cdot \alpha_{\sigma_{i+k}}^{i_{\overline{F}_\epsilon}+k} \cdot z^k}) \\ \Leftrightarrow \quad & e(g, \widetilde{g})^{\overline{a}_\epsilon \cdot v_{i_{\overline{F}_\epsilon}} \cdot \sum_{k=0}^{l-1} \alpha'_{\sigma_{i+k}} \cdot (\alpha_{\sigma_{i+k}} \cdot z)^{(i_{\overline{F}_\epsilon}+k)}} \\ & \qquad = e(g, \widetilde{g})^{\overline{a}_\epsilon \cdot v_{i_{\overline{F}_\epsilon}} \cdot \sum_{k=0}^{l-1} \alpha'_{\sigma_{w,k}} \cdot (\alpha_{\sigma_{w,k}} \cdot z)^{(i_{\overline{F}_\epsilon}+k)}} \end{aligned} \tag{1}$$

$$\Leftrightarrow \quad \sum_{k=0}^{l-1} \alpha'_{\sigma_{i+k}} \cdot (\alpha_{\sigma_{i+k}} \cdot z)^{(i_{\overline{F}_\epsilon}+k)} = \sum_{k=0}^{l-1} \alpha'_{\sigma_{w,k}} \cdot (\alpha_{\sigma_{w,k}} \cdot z)^{(i_{\overline{F}_\epsilon}+k)}$$

$$\Leftrightarrow \quad \sum_{k \in \mathcal{K}_{\neq}} (\alpha'_{\sigma_{i+k}} \cdot \alpha_{\sigma_{i+k}}^{i_{\overline{F}}+k} - \alpha'_{\sigma_{w,k}} \cdot \alpha_{\sigma_{w,k}}^{i_{\overline{F}}+k}) \cdot z^{i_{\overline{F}}+k} = 0$$

Since $\sigma_{w,k} \neq \sigma_{i+k}, \forall k \in \mathcal{K}_{\neq}$, then the probability that the previous equation holds is equivalent to the probability that a random scalar z ($z \xleftarrow{\$} \mathbb{Z}_p$) is a root of a non-zero polynomial of degree at most $l-1$. As a result, the probability that **Test** returns a false positive is at most $\frac{l-1}{p}$ which is negligible (since p is a large prime).

– **Case 2:** $i \in F_\epsilon \backslash \overline{F}_\epsilon$ This can be proved using the same strategy as in the previous case. One needs just to replace $\overline{C}_i$ by C_i, $\overline{C'}_i$ by C'_i, and $i_{\overline{F}_\epsilon}$ by i_{F_ϵ}.

A.2 Proof of Theorem 2

To prove the trace indistinguishability property of our construction, we use the same strategy as in [1]. Let G_0^β be the $Exp_{\mathcal{A},\beta}^{T\text{-}IND\text{-}CPA}$ as define in Definition 1. We will use a sequence of games $G_j^{(\beta)}$ for $j \in [1,n]$ to show that the advantage of the adversary $\mathcal{A}$ for winning $Exp_{\mathcal{A},\beta}^{T\text{-}IND\text{-}CPA}$ is negligible.

Let us suppose that $T_0 = \sigma_{0,1}^* \cdots \sigma_{0,m-1}^*$ and $T_1 = \sigma_{1,1}^* \cdots \sigma_{1,m-1}^*$ are the two traces chosen by $\mathcal{A}$ in $Exp_{\mathcal{A},\beta}^{T\text{-}IND\text{-}CPA}$ (Definition 1). We denote by $\mathcal{I}_{\neq}$ the set of indexes j in which $\sigma_{0,i}^* \neq \sigma_{1,i}^*$ and by $\mathcal{I}_{\neq}^{(j)}$ the subset containing the first j indexes of $\mathcal{I}_{\neq}$ (if $j < |\mathcal{I}_{\neq}|$, then $\mathcal{I}_{\neq}^{(j)} = \mathcal{I}_{\neq}$). In this proof, we rely on a standard hybrid argument in which an element of the challenge ciphertext is randomized at each game hop. That is, for $j \in [1,n]$, the game $G_j^{(\beta)}$ modifies $G_0^{(\beta)}$ by changing, for all $i \in \mathcal{I}_{\neq}^{(j)}$, the element C_i and $\overline{C}_i$(if $\overline{C}_i \neq$ Null) of the challenge ciphertext to random elements of $\mathbb{G}_1$. This means that the last game G_n^β, the challenge cipertext does not contain any useful information about $\sigma_{\beta,i} \forall i \in \mathcal{I}_{\neq}$. As a result, the adversary cannot distinguish whether it plays $G_n^{(0)}$ or $G_n^{(1)}$.

In [1], authors showed that $Adv^{Exp_{\mathcal{A},\beta}^{T\text{-}IND\text{-}CPA}}(\lambda)$ can be bounded as following:

$$Adv^{Exp_{\mathcal{A},\beta}^{T\text{-}IND\text{-}CPA}}(\lambda) \leq \sum_{j=1}^{n-1} |G_j^{(1)}(\lambda) - G_{j+1}^{(1)}(\lambda)| + \sum_{j=1}^{n-1} |G_{j+1}^{(0)}(\lambda) - G_j^{(0)}(\lambda)|$$

Therefore, in order to show that $Adv^{Exp_{\mathcal{A},\beta}^{T\text{-}IND\text{-}CPA}}(\lambda)$ is negligible, we need to show that for all $j \in [0, n-1]$, for all $\beta \in \{0,1\}$, $|Pr[G_j^\beta(\lambda) = 1] - Pr[G_{j+1}^\beta(\lambda) = 1]|$ is negligible. This is stated by the following lemma.

Lemma 1. *For all $j \in [0, n-1]$, for all $\beta \in \{0,1\}$, $|Pr[G_j^\beta(\lambda) = 1] - Pr[G_{j+1}^\beta(\lambda) = 1]|$ is negligible under the i-GDH assumption for $S = T = \emptyset$, $R = \{z^i, a_k \cdot z^i, \overline{a}_k \cdot z^i\}_{i=0,k=0}^{i=2n-1,k=nf}$, and $f \in \{a_{k^*} \cdot x_0' \cdot x_0^{i^*} \cdot z^n, \overline{a}_{k^*} \cdot x_0' \cdot x_0^{i^*} \cdot z^n, \overline{a}_{k^*+1} \cdot x_0' \cdot x_0^{i^*} \cdot z^n\}$.*

Proof. In this proof, we are mainly considering the case in which $j < |\mathcal{I}_{\neq}|$, since otherwise, $\mathcal{I}_{\neq}^{(j)} = \mathcal{I}_{\neq}^{(j+1)}$. This means that $G_j^\beta = G_{j+1}^\beta$ are exactly the same and there is nothing to prove.

Let i^* be the $(j+1)$st index in $\mathcal{I}_{\neq}$, $\epsilon \in [0, m/fs]$, $g_{i,F} = g^{a_k \cdot z^i}$, and $g_{i,\overline{F}} = g^{\overline{a}_k \cdot z^i}$. From the i-GDH challenge containing $\{g^{z^i}, g^{a_k \cdot z^i}, g^{\overline{a}_k \cdot z^i}\}$ the simulator starts by defining $g^{z^i} = g^{z^{n-i^*+i}}$ and $g_{i,F}$ and $g_{i,\overline{F}}$ according to the following three cases:

- C1.1: $i^* \in \overline{F}_{\epsilon-1}$: The simulator defines $g_{i,\overline{F}} = g^{\overline{a}_{\epsilon-1} \cdot z^{n+i_{\overline{F}_{\epsilon-1}} - i^*_{\overline{F}_{\epsilon-1}}}}$
- C1.2: $i^* \in \overline{F}_{\epsilon}$: The simulator defines $g_{i,\overline{F}} = g^{\overline{a}_{\epsilon} \cdot z^{n+i_{\overline{F}_{\epsilon}} - i^*_{\overline{F}_{\epsilon}}}}$
- C1.3: otherwise ($i^* \in F_\epsilon$): The simulator defines $g_{i,F} = g^{a_\epsilon \cdot z^{n+i_{F_\epsilon} - i^*_{F_\epsilon}}}$

Once the simulator receive an issues query for the pattern $p = \sigma_0^{(p)}, \cdots, \sigma_{l_p-1}^{(p)}$, it start by checking that p satisfies the condition defined in the step (3)(a) of $Exp_{\mathcal{A},\beta}^{T\_IND\_CPA}$ (Definition 1). Then, it uses the simulator $\mathcal{O}^S$ to generate a valid trapdoor for p. One can easily check at this level that $\forall j \in [max(0, i^* + l_p - n), min(i^*, l_p - 1)] : \sigma_0^{(p)}, \cdots, \sigma_{l_p-1}^{(p)} \neq \sigma_{\beta,i^*-j}^* \cdots \sigma_{\beta,i^*-j+l_p-1}^*$. If the previous formula is not satisfied, we end up with

$$\sigma_{1-\beta,i^*-j}^* \cdots \sigma_{1-\beta,i^*-j+l_p-1}^* \neq \sigma_0^{(p)}, \cdots, \sigma_{l_p-1}^{(p)} \neq \sigma_{\beta,i^*-j}^* \cdots \sigma_{\beta,i^*-j+l_p-1}^*$$

which is in contradiction with $i^* \in \mathcal{I}_{\neq}$.

The simulator then creates the challenge $C = \{C_i', C_i, \overline{C'}_i, \overline{C}_i\}_{i=0}^{i=m-1}$ according to the following three cases:

- C2.1: $i \in F_\epsilon \cap \overline{F}_{\epsilon-1}$:
  - $C_i' = g^{a_\epsilon z^{n+i_{F_\epsilon} - i^*_{F_\epsilon}}}$ and $\overline{C'}_i = g^{\overline{a}_{\epsilon-1} z^{n+i_{\overline{F}_{\epsilon-1}} - i^*_{\overline{F}_{\epsilon-1}}}}$
  - $\forall i \in \mathcal{I}^{(j)} : C_i \xleftarrow{\$} \mathbb{G}_1, \overline{C}_i \xleftarrow{\$} \mathbb{G}_1$
  - $\forall i \notin \mathcal{I}^{(j+1)}$ the simulator uses the oracle $\mathcal{O}^R$ to get valid C_i and $\overline{C_i}$ and sets U to be in $\{C_{i^*}, \overline{C}_{i^*}\}$
- C2.2: $i \in F_\epsilon \cap \overline{F}_{\epsilon}$:
  - $C_i' = g^{a_\epsilon z^{n+i_{F_\epsilon} - i^*_{F_\epsilon}}}$ and $\overline{C'}_i = g^{\overline{a}_\epsilon z^{n+i_{\overline{F}_{\epsilon}} - i^*_{\overline{F}_{\epsilon}}}}$
  - $\forall i \in \mathcal{I}^{(j)} : C_i \xleftarrow{\$} \mathbb{G}_1, \overline{C}_i \xleftarrow{\$} \mathbb{G}_1$
  - $\forall i \notin \mathcal{I}^{(j+1)}$ the simulator uses the oracle $\mathcal{O}^R$ to get valid C_i and $\overline{C}_i$ and sets U to be in $\{C_{i^*}, \overline{C}_{i^*}\}$
- C2.3: $i \in F_\epsilon \backslash (\overline{F}_{\epsilon-1} \cup \overline{F}_{\epsilon})$:
  - $C_i' = g^{a_\epsilon z^{n+i_{F_\epsilon} - i^*_{F_\epsilon}}}$ and $\overline{C'}_i = \emptyset$

- $\forall i \in \mathcal{I}^{(j)} : C_i \xleftarrow{\$} \mathbb{G}_1$ and $\overline{C}_i = \emptyset$
- $\forall i \notin \mathcal{I}^{(j+1)}$ the simulator uses the oracle $\mathcal{O}^R$ to get valid C_i and sets $U = C_{i^*}$

Then if

$$U = \begin{cases} \left.\begin{array}{l} \overline{C}_{i^*} = g^{\overline{a}_{\epsilon-1} \cdot \alpha'_{\sigma_{i^*}} \cdot \alpha^{i^*}_{\sigma_{i^*}} \cdot z^n} \\ C_{i^*} = g^{a_\epsilon \cdot \alpha'_{\sigma_{i^*}} \cdot \alpha^{i^*}_{\sigma_{i^*}} \cdot z^n} \end{array}\right\} \text{ or} & \text{if C2.1} \\ \left.\begin{array}{l} \overline{C}_{i^*} = g^{\overline{a}_{\epsilon} \cdot \alpha'_{\sigma_{i^*}} \cdot \alpha^{i^*}_{\sigma_{i^*}} \cdot z^n} \\ C_{i^*} = g^{a_\epsilon \cdot \alpha'_{\sigma_{i^*}} \alpha^{i^*}_{\sigma_{i^*}} \cdot z^n} \end{array}\right\} \text{ or} & \text{if C2.2} \\ C_{i^*} = g^{a_\epsilon \cdot \alpha'_{\sigma_{i^*}} \cdot \alpha^{i^*}_{\sigma_{i^*}} \cdot z^n} & \text{if C2.3} \end{cases}$$

then the simulator is playing the game $G_j^{(\beta)}$. Otherwise U is random and the simulator is playing $G_{j+1}^{(\beta)}$. Then an adversary $\mathcal{A}$ able to distinguish $G_j^{(\beta)}$ and $G_{j+1}^{(\beta)}$ will be able to win $Exp_{\mathcal{A},\beta}^{T\text{-}IND\text{-}CPA}$ with non negligible advantage. By replacing $\alpha_{\sigma_{i^*}}$ by x_0 and $\alpha'_{\sigma_{i^*}}$ by x'_0, in order to prove that $\mathcal{A}$ cannot distinguish $G_j^{(\beta)}$ and $G_{j+1}^{(\beta)}$, we need to prove that for all $f \in \{a_{k^*} \cdot x'_0 \cdot x_0^{i^*} \cdot z^n, \overline{a}_{k^*} \cdot x'_0 \cdot x_0^{i^*} \cdot z^n, \overline{a}_{k^*+1} \cdot x'_0 \cdot x_0^{i^*} \cdot z^n\}$, f is independent of the sets R,S, and T after q queries to $\mathcal{O}^S$ and 1 query to $\mathcal{O}^R$ which will be proved in Lemma 2.

Each pattern $p_t = \sigma_{t,0}, \cdots, \sigma_{t,l_t-1}$ submitted to $\mathcal{O}^S$ will add the polynomials $\sum_{s=0}^{fs-l_t} v_{t,s} \cdot \sum_{k=0}^{l_t-1} \alpha'_{\sigma_{t,k}} (\alpha_{\sigma_{t,k}} \cdot z)^{k+s}$ and $\sum_{s=0}^{fs-l_t} v_{t,s}$ to S. In addition, a query to the oracle $\mathcal{O}^R$ will add $\forall i \in [0, m-1] \backslash \{i^*\}$

$$\begin{cases} \overline{a}_{\epsilon-1} \cdot \alpha'_{\sigma_i^*} \cdot \alpha^i_{\sigma_i^*} \cdot z^{n-i^*+i} & \text{if } i \in F_\epsilon \cap \overline{F}_{\epsilon-1} \\ \overline{a}_{\epsilon} \cdot \alpha'_{\sigma_i^*} \cdot \alpha^i_{\sigma_i^*} \cdot z^{n-i^*+i} & \text{if } i \in F_\epsilon \cap \overline{F}_{\epsilon} \\ a_{\epsilon} \cdot \alpha'_{\sigma_i^*} \cdot \alpha_{\sigma_i^*} \cdot z^{n-i^*+i} & \text{if } i \in F_\epsilon \end{cases} \tag{2}$$

to R.

With this new notations, R initially contains $\{z^i, a_k \cdot z^i, \overline{a}_k \cdot z^i\}_{i=0,k=0}^{i=2n-1,k=nf}$ where T and S are initially empty.

Lemma 2. *Let R,S, and T be the sets defines above after q queries to $\mathcal{O}^S$ and 1 query to $\mathcal{O}^R$. If $\forall t \in [1,q]$, the pattern $p_t = \sigma_{t,0}, \cdots, \sigma_{t,l_t-1}$ submitted to $\mathcal{O}^{iss}$ differs for all $j \in [max(0, i^*+l_t-n), min(i^*, l_t-1)]$ from $\sigma^*_{\beta,i^*-j} \cdots \sigma^*_{\beta,i^*-j+l_t-1}$, then $\forall f \in \{a_{k^*} \cdot \alpha'_{\sigma^*_{i^*}} \cdot \alpha^{i^*}_{\sigma^*_{i^*}} \cdot z^n, \overline{a}_{k^*} \cdot \alpha'_{\sigma^*_{i^*}} \cdot \alpha^{i^*}_{\sigma^*_{i^*}} \cdot z^n, \overline{a}_{k^*+1} \cdot \alpha'_{\sigma^*_{i^*}} \cdot \alpha^{i^*}_{\sigma^*_{i^*}} \cdot z^n\}$, f is independent of $< R, S, T >$.*

Proof. According to Definition 5, to prove that f is independent of $< R, S, T >$, we need to prove that $\forall a \in \{a_{k^*}, \overline{a}_{k^*}, \overline{a}_{k^*+1}\}$, there is no combination of polynomials from R,S, and T such that

$$\begin{aligned} \left(a \cdot \alpha'_{\sigma^*_{i^*}} \cdot \alpha^{i^*}_{\sigma^*_{i^*}} \cdot z^n\right) \left(\sum_j u_j^a \cdot S^{(j)}\right) = \\ \sum_{i,j} u_{i,j}^b \cdot R^{(i)} \cdot S^{(j)} + \sum_k u_k^{(c)} T^{(t)} \end{aligned} \tag{3}$$

First, we remark that the factor a appears only in the elements $\{a_k \cdot z^i, \overline{a}_k \cdot z^i\}_{i=0,k=0}^{i=2n-1,k=nf}$ of R and in the output of the oracle $\mathcal{O}^R$ (Formula 2). Thus, the elements of R that are not multiple of a cannot be part of Equation (3). In addition, the last sum of Equation (3) can be removed since T is empty. So, let $\{\vartheta_{i,t,s}^{(a)}, \vartheta_{i,t,s}^{(b)}, \vartheta_j^{(c)}, \vartheta_{j,t,s}^{(d)}, \vartheta_{j,t,s}^{(e)}, \vartheta_i^{(f)}\}_{i=0,j=0,t=1,s=0}^{i=\delta-1,j=2n-1,t=q,s=fs-l_t}$ be constants such that

$$
\begin{aligned}
& a \cdot \alpha'_{\sigma_{i^*}^*} \cdot \alpha_{\sigma_{i^*}^*}^{i^*} \cdot z^n (\sum_{t=1}^{q} \sum_{s=0}^{fs-l_t} (\vartheta_{i^*,t,s}^{(a)} \cdot V_{t,s} + \vartheta_{i^*,t,s}^{(b)} \cdot v_{t,s})) \\
& = \sum_{j=0}^{2n-1} a \cdot \vartheta_j^{(c)} \cdot z^j (\sum_{t=1}^{q} \sum_{s=0}^{fs-l_t} (\vartheta_{j,t,s}^{(d)} \cdot V_{t,s} + \vartheta_{j,t,s}^{(e)} \cdot v_{t,s})) \\
& + (\sum_{i=0,i\neq i^*}^{\delta-1} a \cdot \vartheta_i^{(f)} \cdot \alpha'_{\sigma_i^*} \cdot \alpha_{\sigma_i^*}^{i} \cdot z^{n-i^*+i}) \cdot \\
& \qquad (\sum_{t=1}^{q} \sum_{s=0}^{fs-l_t} (\vartheta_{i,t,s}^{(a)} \cdot V_{t,s} + \vartheta_{i,t,s}^{(b)} \cdot v_{t,s}))
\end{aligned}
$$

where $\delta = fs$ if $a = a_{k^*}$ (i.e, there are at most fs elements in the fragments in which a_{k^*} is used) and $\delta = 2p_{max} - 2$ if $a = \overline{a}_{k^*}$ or $a = \overline{a}_{k^*+1}$ (i.e, there are at most $2p_{max} - 2$ elements in the fragments in which $\overline{a}_{k^*}$ is used).

Our goal is then to show that $\vartheta_{i^*,t,s}^{(a)} = \vartheta_{i^*,t,s}^{(b)} = 0$ for any $s \in [0, fs - l_t]$ and $t \in [1, q]$. Let us consider each member of the previous equation as a polynomial in the variable $\{\alpha'_\sigma\}_{\sigma \in \Sigma}$. We regroup the different monomials according to their degree and we divide each member by a:

1. $\sum_{j=0}^{2n-1} \vartheta_j^{(c)} \cdot z^j (\sum_{t=1}^{q} \sum_{s=0}^{fs-l_t} \vartheta_{j,t,s}^{(e)} \cdot v_{t,s}) = 0$
2. $\alpha'_{\sigma_{i^*}^*} \cdot \alpha_{\sigma_{i^*}^*}^{i^*} \cdot z^n (\sum_{t=1}^{q} \sum_{s=0}^{fs-l_t} \vartheta_{i^*,t,s}^{(b)} \cdot v_{t,s}) = \sum_{j=0}^{2n-1} \vartheta_j^{(c)} \cdot z^j (\sum_{t=1}^{q} \sum_{s=0}^{fs-l_t} \vartheta_{j,t,s}^{(d)} \cdot V_{t,s}) + (\sum_{i=0,i\neq i^*}^{\delta-1} \vartheta_i^{(f)} \cdot \alpha'_{\sigma_i^*} \cdot \alpha_{\sigma_i^*}^{i} \cdot z^{n-i^*+i}) (\sum_{t=1}^{q} \sum_{s=0}^{fs-l_t} \vartheta_{i,t,s}^{(b)} \cdot v_{t,s})$
3. $\alpha'_{\sigma_{i^*}^*} \cdot \alpha_{\sigma_{i^*}^*}^{i^*} \cdot z^n (\sum_{t=1}^{q} \sum_{s=0}^{fs-l_t} \vartheta_{i^*,t,s}^{(a)} \cdot V_{t,s}) = (\sum_{i=0,i\neq i^*}^{\delta-1} \vartheta^{(f)_i} \cdot \alpha'_{\sigma_i^*} \cdot \alpha_{\sigma_i^*}^{i} \cdot z^{n-i^*+i}) (\sum_{t=1}^{q} \sum_{s=0}^{fs-l_t} \vartheta_{i,t,s}^{(a)} \cdot V_{t,s})$

In Equation (3), since $V_{t,s} = v_{t,s} \cdot \sum_{k=0}^{l_t-1} \alpha'_{\sigma_{t,i}} \alpha_{\sigma_{t,i}}^{s+k} \cdot z^k$ and $\forall t, t' \in [1, q], \forall s, s' \in [0, fs - l_t] : v_{t,s} \neq v_{t',s'}$ with overwhelming probability (in the Issues algorithm (Algorithm 2), $v_{t,s}$ is chosen randomly from $\mathbb{Z}_p$), we have $\forall t \in [1, q], \forall s \in [0, fs - l_t]$:

$$
\begin{aligned}
& \alpha'_{\sigma_{i^*}^*} \cdot \alpha_{\sigma_{i^*}^*}^{i^*} \cdot z^n \cdot \vartheta_{i^*,t,s}^{(a)} \cdot V_{t,s} = \\
& \sum_{i=0,i\neq i^*}^{\delta-1} \vartheta^{(f)_i} \cdot \alpha'_{\sigma_i^*} \cdot \alpha_{\sigma_i^*}^{i} \cdot z^{n-i^*+i} \cdot \vartheta_{i,t,s}^{(a)} \cdot V_{t,s}
\end{aligned}
$$

We can remove $V_{t,s}$ in each member of the last equation to get:

$$
\alpha'_{\sigma_{i^*}} \cdot \alpha_{\sigma_{i^*}}^{i^*} \cdot z^n \cdot \vartheta_{i^*,t,s}^{(a)} = \sum_{i=0,i\neq i^*}^{\delta-1} \vartheta^{(f)_i} \cdot \alpha'_{\sigma_i^*} \cdot \alpha_{\sigma_i^*}^{i} \cdot z^{n-i^*+i} \cdot \vartheta_{i,t,s}^{(a)}
$$

We note that we cannot get a monomial of degree n in z in the right member of the last equation which means that $\vartheta^{(a)}_{i^*,t,s} = 0$, $\forall t \in [1,q]$ and $\forall s \in [0, fs - l_t]$. It then only remains to prove that $\forall t \in [1,q], \forall s \in [0, fs - l_t] : \vartheta^{(b)}_{i^*,t,s} = 0$.

In Equation (2), let us define $\vartheta^{(f)}_{i^*} = -1$. Since $\forall t, t' \in [1,q], \forall s, s' \in [0, fs - l_t] : v_{t,s} \neq v_{t',s'}$ with overwhelming probability, we can merge the left member with the last sum of the right member to get $\forall t \in [1,q], \forall s \in [0, fs - l_t]$:

$$\sum_{j=0}^{2n-1} \vartheta^{(c)}_j \cdot z^j \cdot \vartheta^{(d)}_{j,t,s} \cdot V_{t,s} =$$
$$-(\sum_{i=0}^{\delta-1} \vartheta^{(f)}_i \cdot \alpha'_{\sigma^*_i} \cdot \alpha^i_{\sigma^*_i} \cdot z^{n-i^*+i})(\vartheta^{(b)}_{i,t,s} \cdot v_{t,s})$$

Now, by replacing $V_{t,s}$ by $v_{t,s} \cdot \sum_{k=0}^{l_t-1} \alpha'_{\sigma_{t,k}} \alpha^{s+k}_{\sigma_{t,k}} \cdot z^k$ we have:

$$\sum_{j=0}^{2n-1} \vartheta^{(c)}_j \cdot z^j \cdot \vartheta^{(d)}_{j,t,s} \cdot \sum_{k=0}^{l_t-1} \alpha'_{\sigma_{t,k}} \alpha^{s+k}_{\sigma_{t,k}} \cdot z^k =$$
$$-\sum_{i=0}^{\delta-1} \vartheta^{(f)}_i \cdot \alpha'_{\sigma^*_i} \cdot \alpha^i_{\sigma^*_i} \cdot z^{n-i^*+i} \cdot \vartheta^{(b)}_{i,t,s}$$

By regrouping z elements in the left side we get:

$$\sum_{j=0}^{2n+l_t-2} z^j \sum_{k=0}^{l_t-1} \vartheta^{(c)}_{j-k} \cdot \vartheta^{(d)}_{j-k,t,s} \cdot \alpha'_{\sigma_{t,k}} \cdot \alpha^{s+k}_{\sigma_{t,k}} =$$
$$-\sum_{i=0}^{\delta-1} \vartheta^{(f)}_i \cdot \alpha'_{\sigma^*_i} \cdot \alpha^i_{\sigma^*_i} \cdot z^{n-i^*+i} \cdot \vartheta^{(b)}_{i,t,s} \tag{4}$$

where $\vartheta^{(c)}_{i-k} = \vartheta^{(d)}_{i,t,s} = 0$ if $i \geq 2n$. So if we consider the monomial of degree n in z we get: $\sum_{k=0}^{l_t-1} \vartheta^{(c)}_{n-k} \cdot \vartheta^{(d)}_{n-k,t,s} \cdot \alpha'_{\sigma_{t,k}} \cdot \alpha^{s+k}_{\sigma_{t,k}} = \vartheta^{(f)}_{i^*} \cdot \alpha'_{\sigma^*_{i^*}} \cdot \alpha^{i^*}_{\sigma^*_{i^*}} \cdot \vartheta^{(b)}_{i^*,t,s}$. Since by definition $\vartheta^{(f)}_{i^*} = -1$, then to show that $\vartheta^{(b)}_{i^*,t,s}$, we will show that $\vartheta^{(c)}_{n-k} \cdot \vartheta^{(d)}_{n-k,t,s} = 0$ for all $k \in [0, l_t - 1]$.

By definition, we have for all $j \in [max(0, i^* + l_t - n), min(i^*, l_t - 1)]$: $\sigma_{t,0}, \cdots, \sigma_{t,l_t-1} \neq \sigma^*_{\beta,i^*-j} \cdots \sigma^*_{\beta,i^*-j+l_t-1}$. Thus, $\forall i \in [max(0, i^* - l_t + 1), min(\delta - l_t, i^*)], \exists \bar{i} \in [0, l_t - 1]$ such that $\sigma^*_{i+\bar{i}} \neq \sigma_{\bar{i}}$. Let us now consider the coefficient associated with the monomial of degree $n - i^* + i + \bar{i}$ in z. Then $\forall i \in [max(0, i^* - l_t + 1), min(\delta - l_t, i^*)]$ we have:

$$\sum_{k=0}^{l_t-1} \vartheta^{(c)}_{n-i^*+i+\bar{i}-k} \cdot \vartheta^{(d)}_{n-i^*+i+\bar{i}-k,t,s} \cdot \alpha'_{\sigma_{t,k}} \cdot \alpha^{s+k}_{\sigma_{t,k}} =$$
$$\vartheta^{(f)}_{i+\bar{i}} \cdot \alpha'_{\sigma^*_{i+\bar{i}}} \cdot \alpha^{i+\bar{i}}_{\sigma^*_{i+\bar{i}}} \cdot \vartheta^{(b)}_{i+\bar{i},t,s}$$

Since $\sigma_{t,\bar{i}} \neq \sigma^*_{i+\bar{i}} \Leftrightarrow \alpha_{\sigma_{t,\bar{i}}} \neq \alpha_{\sigma^*_{i+\bar{i}}}$, we have:

$$\sum_{k=0,\sigma_{t,k}=\sigma_{t,\bar{i}}}^{l_t-1} \vartheta^{(c)}_{n-i^*+i+\bar{i}-k} \cdot \vartheta^{(d)}_{n-i^*+i+\bar{i}-k,t,s} \cdot \alpha'_{\sigma_{t,k}} \cdot \alpha^{s+k}_{\sigma_{t,k}} = 0$$

which means that $\vartheta^{(c)}_{n-i^*+i+\bar{i}-k} \cdot \vartheta^{(d)}_{n-i^*+i+\bar{i}-k,t,s}$ for all k such that $\sigma_{t,k} = \sigma_{t,\bar{i}}$, and in particular $k = \bar{i}$. This means that $\vartheta^{(c)}_{n-i^*+i} \cdot \vartheta^{(d)}_{n-i^*+i,t,s} = 0$ for all $i \in [max(0, i^* - l_t + 1), min(\delta - l_t, i^*)]$, which implies that $\vartheta^{(c)}_{n-k} \cdot \vartheta^{(d)}_{n-k,t,s} = 0$ for all $k \in [i^* - min(\delta - l_t, i^*), i^* - max(0, i^* - l_t + 1)]$.
As a result, we have:

- If $min(i^*, \delta - l_t) = i^*$ and $max(0, i^* - l_t + 1) = i^* - l_t + 1$ then $\vartheta^{(c)}_{n-k} \cdot \vartheta^{(d)}_{n-k,t,s} = 0$ for all $k \in [0, l_t - 1]$ which equivalent to $\vartheta^{(b)}_{i^*,t,s} = 0$, and thus the independence of $a \cdot \alpha'_{\sigma^*_{i^*}} \cdot \alpha^{i^*}_{\sigma^*_{i^*}} \cdot z^n$.
- If $i^* > \delta - l_t$: In this case we must prove that $\vartheta^{(c)}_{n-k} \cdot \vartheta^{(d)}_{n-k,t,s} = 0$ for any $k \in [0, i^* + l_t - \delta - 1]$. Proof is contradiction. So, let us assume that there is $\overline{k} \in [0, i^* + l_t - \delta - 1]$ such that $\vartheta^{(c)}_{n-\overline{k}} \cdot \vartheta^{(d)}_{n-\overline{k},t,s} \neq 0$. So, let us consider the monomials of degree $n - \overline{k} + l_t - 1$ in z of Equation (4). The coefficient of its left member is $\sum_{k=0}^{l_t-1} \vartheta^{(c)}_{n-\overline{k}+l_t-1-k} \cdot \vartheta^{(d)}_{n-\overline{k}+l_t-1-k,t,s} \cdot \alpha'_{\sigma_{t,k}} \cdot \alpha^k_{\sigma_{t,k}}$ and in the right member is 0, since the degree of the right member is at most $n + \delta - i^* - 1$. Or $\overline{k} \leq i^* + l_t - \delta - 1 \Leftrightarrow n - \overline{k} + l_t - 1 \geq n - i^* + \delta$. So,

  $$\sum_{k=0}^{l_t-1} \vartheta^{(c)}_{n-\overline{k}+l_t-1-k} \cdot \vartheta^{(d)}_{n-\overline{k}+l_t-1-k,t,s} \cdot \alpha'_{\sigma_{t,k}} \cdot \alpha^{s+k}_{\sigma_{t,k}} = 0.$$

  which means that $\vartheta^{(c)}_{n-\overline{k}+l_t-1-k} \cdot \vartheta^{(d)}_{n-\overline{k}+l_t-1-k,t,s} = 0$ for all $k \in [0, l_t - 1]$ and in particular $k = l_t - 1$. However, this contradicts our assumption $\vartheta^{(c)}_{n-\overline{k}} \cdot \vartheta^{(d)}_{n-\overline{k},t,s} \neq 0$. Thus, $\vartheta^{(c)}_{n-k} \cdot \vartheta^{(d)}_{n-k,t,s} = 0$ for any $k \in [0, i^* + l_t - \delta - 1]$.
- If $i^* < l_t - 1$: Here we must prove that $\vartheta^{(c)}_{n-k} \cdot \vartheta^{(d)}_{n-k,t,s} = 0$ for any $k \in [i^* + 1, l_t - 1]$. Proof is by contradiction. We again assume that $\exists \overline{k} \in [i^* + 1, l_t - 1]$ such that $\vartheta^{(c)}_{n-\overline{k}} \cdot \vartheta^{(d)}_{n-\overline{k},t,s} \neq 0$. Let us consider the monomials of degree $n - \overline{k}$ in z of Equation (4). The coefficient of its left member is $\sum_{k=0}^{l_t-1} \vartheta^{(c)}_{n-\overline{k}-k} \cdot \vartheta^{(d)}_{n-\overline{k}-k,t,s} \cdot \alpha'_{\sigma_{t,k}} \cdot \alpha^k_{\sigma_{t,k}}$ and in the right member is 0, since the degree of the right member is at least $n - i^*$ and $\overline{k} > i^* + 1 \Leftrightarrow n - \overline{k} \leq n - i^* - 1$. Therefore we have:

  $$\sum_{k=0}^{l_t-1} \vartheta^{(c)}_{n-\overline{k}-k} \cdot \vartheta^{(d)}_{n-\overline{k}-k,t,s} \cdot \alpha'_{\sigma_{t,k}} \cdot \alpha^{s+k}_{\sigma_{t,k}} = 0.$$

  which means that $\vartheta^{(c)}_{n-\overline{k}-k} \cdot \vartheta^{(d)}_{n-\overline{k}-k,t,s} = 0$ for all $k \in [0, l_t - 1]$ and in particular $k = 0$. However, this contradicts our assumption $\vartheta^{(c)}_{n-\overline{k}} \cdot \vartheta^{(d)}_{n-\overline{k},t,s} \neq$

0. Thus, $\vartheta^{(c)}_{n-k} \cdot \vartheta^{(d)}_{n-k,t,s} = 0$ for any $k \in [i^* + 1, l_t - 1]$, which conclude the proof.

A.3 Proof of Theorem 3

According to Theorem 2, despite the fact that the adversary $\mathcal{A}$ is able to adaptively query the $\mathcal{O}^S$ to issue trapdoors for a finite set of patterns $\mathcal{P}$, our construction is proved to be trace indistinguishable. This means that, in $Exp^{ETF}_{\mathcal{A}}$ game, the adversary will not be able to get any information about the ciphertext C^* of the trace T^* out of $C^T, T \in \mathcal{T}$ (since $T^* \notin \mathcal{T}$).

So let us suppose that $T^* = \sigma_{T^*,0} \cdots \sigma_{T^*,n-1}$, $w_t = \sigma_{w_t,0} \cdots \sigma_{w_t,l-1}$, and that $\mathcal{A}$ forges C^* using the key $K^* = \{z^*, \{\alpha^*_\sigma, \alpha'^*_\sigma\}_{\sigma \in \Sigma}\}$ in such a way that $\exists i \in [0, |T| - 1] : i \in Test(C^*, td_{w_t})$ and that $td_{w_t} = \{V_j, v_j\}_{j=0}^{j=fs-l}$. Again, two cases should be considered:

- **Case 1:** $\epsilon < nf - 1$ **and** $i \in F_\epsilon \cap \overline{F}_\epsilon$

$$e(\prod_{j=0}^{l-1} \overline{C^*}_{i+j}, \widetilde{g}^{v_{i_{\overline{F}_\epsilon}}}) = e(\overline{C^{*\prime}}_i, \widetilde{g}^{V_{i_{\overline{F}_\epsilon}}})$$

  By using the same transformation as in the proof of Theorem 1, we get

$$\sum_{k=0}^{l-1} \alpha'^*_{\sigma_{T^*,i+k}} \cdot (\alpha^*_{\sigma_{T^*,i+k}} \cdot z^*)^{(i_{\overline{F}_\epsilon}+k)} = \sum_{k=0}^{l-1} \alpha'_{\sigma_{w_t,k}} \cdot (\alpha_{\sigma_{w_t,k}} \cdot z)^{(i_{\overline{F}_\epsilon}+k)}$$

  Since, T^* contains w_t at index i, then $\forall k \in [0, l-1], \sigma_{T^*,i+k} = \sigma_{w_t,k}$ The previous equation only holds if $z = z^*$, $\alpha'^*_{\sigma_{w_t,k}} = \alpha'_{\sigma_{w_t,k}}$, and $\alpha_{\sigma_{w_t,k}} = \alpha_{\sigma_{w_t,k}}$. Since $z \xleftarrow{\$} \mathbb{Z}_p$, $\forall \sigma \in \Sigma : \alpha_\sigma \xleftarrow{\$} \mathbb{Z}_p$ and $\alpha'_\sigma \xleftarrow{\$} \mathbb{Z}_p$, and $z, \alpha_\sigma, \alpha'_\sigma$ are not known to $\mathcal{A}$, then the probability that the adversary $\mathcal{A}$ to win $Exp^{ETF}_{\mathcal{A}}$ is at most $\frac{1}{p^3}$ which is negligible.
- **Case 2:** $i \in F_\epsilon \backslash \overline{F}_\epsilon$: we use the same strategy as in case 1 to show that the advantage of $\mathcal{A}$ to win $Exp^{ETF}_{\mathcal{A}}$ is at most $\frac{1}{p^3}$ which is negligible.

A.4 Proof of Theorem 4

As defined in Definition 3, to show that our construction is pattern indistinguishable, we need to show that the advantage of the adversary $\mathcal{A}$ of winning the game $Exp^{P\_IND\_CPA}_{\mathcal{A},\beta}$, is negligible. So, let $\mathcal{T}$ be the set of (unknown) trace ciphertexts observed in the step 2 of the game $Exp^{P\_IND\_CPA}_{\mathcal{A},\beta}$. Let us first note that since the adversary $\mathcal{A}$ will not have the ability to create valid encrypted traces of his choice (as we showed in Theorem 3), $\mathcal{A}$ will not be able to brute

force the trapdoors by creating a lot of (random) traffics to guess the logic behind them. In addition, according to the Theorem 2, our construction is trace indistinguishable, this means that, since $\forall T \in \mathcal{T}, w^*_\beta \notin T$. $\mathcal{A}$ will not learn any information out of the encrypted traces $\mathcal{T}$, and therefore, the observation of $\mathcal{T}$ will not give $\mathcal{A}$ any advantage in guessing β. As a result, the only solution left to $\mathcal{A}$ is to use the trapdoors provided by $\mathcal{O}^S$ in the query phase of $Exp^{P\_IND\_CPA}_{\mathcal{A},\beta}$. In the following we will show that guessing the pattern w^*_β out of the adaptively chosen patterns w_i and their issued trapdoors td_{w_i} is hard under the i-GDH assumption.

Let fs be the size of fragment we will used in our construction. Suppose that the two challenge patterns chosen by $\mathcal{A}$ are $w^*_0 = \sigma^*_{0,0} \cdots \sigma^*_{0,l-1}$ and $w^*_1 = \sigma^*_{1,0} \cdots \sigma^*_{1,l-1}$. Let $G^{(\beta)}_0$ denotes the $Exp^{P\_IND\_CPA}_{\mathcal{A},\beta}$ game, we will use a sequence of games $G^{(\beta)}_j, j \in [0, fs-1]$ to show that $\mathcal{A}$'s advantage is negligible. As in the proof of Theorem 2, we rely on a standard hybrid argument in which an element of the challenge trapdoor is randomized at each game hop. That is, for $j \in [1, fs-1]$, the game $G^{(\beta)}_j$ modifies $G^{(\beta)}_0$ by changing, for all $i \in [0, j]$, the element V_i of the challenge trapdoor to a random element of $\mathbb{G}_2$. This means that the last game G^{β}_{fs-1}, the challenge trapdoor does not contain any useful information about w^*_β. As a result, the adversary cannot distinguish whether it plays $G^{(0)}_{fs-1}$ or $G^{(1)}_{fs-1}$. As a result, we can bound the advantage of $\mathcal{A}$ as following:

$$Adv^{Exp^{P\_IND\_CPA}_{\mathcal{A},\beta}}(\lambda) \leq$$
$$\sum_{j=1}^{fs-1} |G^{(1)}_j(\lambda) - G^{(1)}_{j+1}(\lambda)| + \sum_{j=1}^{fs-1} |G^{(0)}_{j+1}(\lambda) - G^{(0)}_j(\lambda)|$$

Then to prove that $Adv^{Exp^{P\_IND\_CPA}_{\mathcal{A},\beta}}(\lambda)$ is negligible, we should prove that $\mathcal{A}$ cannot distinguish $G^{(\beta)}_j$ and $G^{(\beta)}_{j+1}$ which is stated by the following Lemma. Assuming the following lemma is proved, each term above is negligible under the i-GDH assumption, which concludes the proof.

Lemma 3. *After performing q queries to $\mathcal{O}^S$, for $j \in [0, fs-1], \beta \in \{0,1\}$, $|Adv^{G^{(\beta)}_j}(\lambda) - Adv^{G^{(\beta)}_{j+1}}(\lambda)|$ is negligible under the i-GDH assumption where $f = v^*_{j+1} \cdot \sum_{k=0}^{l-1} \alpha'_{\sigma^*_{\beta,k}} \cdot \alpha^{j+k+1}_{\sigma^*_{\beta,k}} \cdot z^k$, $R = \{v_{t,s} \cdot \sum_{k=0}^{l-1} \alpha'_{\sigma_{t,k}} \cdot \alpha^{s+k}_{\sigma_{t,k}} \cdot z^k, v_{t,s}, v^*_s \cdot \sum_{k=0}^{l-1} \alpha'_{\sigma^*_k} \cdot \alpha^{s+k}_{\sigma^*_k} \cdot z^k, v^*_s\}^{t=q,s'=s=fs-l}_{t=1,s=0,s'=0,s'\neq j+1}$, $S = \{z^i\}^{i=fs-1}_{i=0}$, and $T = \emptyset$.*

Proof. According to the standard hybrid argument strategy we described before, in each G^{β}_{j+1}, to answer $\mathcal{A}$'s challenge the simulator uses the oracle $\mathcal{O}^S$ to get a valid trapdoor $td_{w^*_\beta} = \{\widetilde{g}^{V^*_s}, \widetilde{g}^{v^*_s}\}^{fs-l}_{s=0}$ for w^*_β. It replaces $\widetilde{g}^{V^*_i}, i \in [0, j]$ by random elements of $\mathbb{G}_2$ and sets $\widetilde{g}^{V^*_{j+1}}$ as U. Then, if $U = v^*_{j+1} \cdot \sum_{k=0}^{l-1} \alpha'_{\sigma^*_{\beta,k}} \cdot \alpha^{j+k+1}_{\sigma^*_{\beta,k}} \cdot z^k$ then the simulator is playing $G^{(\beta)}_j$. Otherwise, U is random and the simulator is playing the $G^{(\beta)}_{j+1}$. Then if $\mathcal{A}$ is able to distinguish $G^{(\beta)}_j$ and $G^{(\beta)}_{j+1}$ he/she will be able to win $Exp^{P\_IND\_CPA}_{\mathcal{A},\beta}$ with non negligible advantage. According to

Definition 6, in order to prove that $\mathcal{A}$ cannot distinguish $G_j^{(\beta)}$ and $G_{j+1}^{(\beta)}$ under i-GDH assumption, we need to prove that for $f = v_{j+1}^* \cdot \sum_{k=0}^{l-1} \alpha'_{\sigma^*_{\beta,k}} \cdot \alpha^{j+k+1}_{\sigma^*_{\beta,k}} \cdot z^k$ is independent of the sets R,S, and T after q queries to $\mathcal{O}^S$.

First let us note that each query issued to $\mathcal{O}^S$ and associated which the pattern $w_i = \sigma_{i,0}, \cdots \sigma_{i,l-1}$, adds, according to the Issue algorithm of our construction, $\{v_{t,s} \cdot \sum_{k=0}^{l-1} \alpha'_{\sigma_{t,k}} \cdot \alpha^{s+k}_{\sigma_{t,k}} \cdot z^k, v_{t,s}\}_{t=1,s=0}^{t=q,s=fs-l}$ to the set R. Moreover, the challenge query adds $\{v^*_{t,s} \cdot \sum_{k=0}^{l-1} \alpha'_{\sigma^*_{t,k}} \cdot \alpha^{s+k}_{\sigma^*_{t,k}} \cdot z^k, v^*_{t,s'}\}_{t=1,s=0,s\neq j+1,s'=0}^{t=q,s=fs-l,s'=fs-l}$.

As we mentioned before, $\mathcal{A}$ will not be able to create a valid ciphertext for chosen trace, then the set S will contain only the elements of $\mathbb{G}_1$ that are provided in *params*. Therefore $S = \{z^i\}_{i=0}^{i=fs-1}$.

Consequently, R contains $\{v_{t,s} \cdot \sum_{k=0}^{l-1} \alpha'_{\sigma_{t,k}} \cdot \alpha^{s+k}_{\sigma_{t,k}} \cdot z^k, v_{t,s}, v^*_s \cdot \sum_{k=0}^{l-1} \alpha'_{\sigma^*_k} \cdot \alpha^{s+k}_{\sigma^*_k} \cdot z^k, v^*_s\}_{t=1,s=0,s'=0,s'\neq j+1}^{t=q,s'=s=fs-l}$, S contains $\{z^i\}_{i=0}^{i=fs-1}$, and T is empty.

So, according to Definition 5, the goal is to prove that one cannot find a combination of polynomials from R, S and T such that:

$$(v^*_{j+1} \cdot \sum_{k=0}^{l-1} \alpha'_{\sigma^*_{\beta,k}} \cdot \alpha^{j+k+1}_{\sigma^*_{\beta,k}} \cdot z^k)(\sum_j u^a_j \cdot S^{(j)}) = \sum_{i,j} u^b_{i,j} \cdot R^{(i)} \cdot S^{(j)} + \sum_k u^{(c)}_k T^{(t)} \tag{5}$$

First let us note that the factor v^*_{j+1} only appears in the last element of the set R. Since $\forall t_1, t_2 \in [1, q], \forall s_1, s_2 \in [0, fs-1]$, $v_{t_1,s_1} \neq v_{t_2,s_2} \neq v^*_{s_1}$ with overwhelming probability, then only the element v^*_{j+1} of the last element of R will be involved in Equation 5. Moreover, since T is empty, the last sum of the Equation 5 can be omitted. Let $\{\vartheta^{(a)}_i, \vartheta^{(a)}_i\}_{i=0}^{i=fs-1}$ be constant scalars such that

$$(v^*_{j+1} \cdot \sum_{k=0}^{l-1} \alpha'_{\sigma^*_{\beta,k}} \cdot \alpha^{j+k+1}_{\sigma^*_{\beta,k}} \cdot z^k)(\sum_{i=0}^{fs-1} \vartheta^{(a)}_i \cdot z^i) = v^*_{j+1}(\sum_{i=0}^{fs-1} \vartheta^{(b)}_i \cdot z^i) \tag{6}$$

So, in order to prove the independence of $v^*_{j+1} \cdot \sum_{k=0}^{l-1} \alpha'_{\sigma^*_{\beta,k}} \cdot \alpha^{j+k+1}_{\sigma^*_{\beta,k}} \cdot z^k$ we must prove that $\forall i \in [0, fs-1], \vartheta^{(a)}_i = 0$. For that reason, let us consider the monomial of degree $j+1$ in $\alpha_{\sigma^*_{\beta,k}}$. In the left member of Equation 6, the coefficient is $\sum_{k=0}^{l-1} \alpha'_{\sigma^*_{\beta,k}} \cdot \sum_{i=0}^{fs-1} \vartheta^{(a)}_i \cdot z^i$ and in its right member the coefficient is 0. Therefore, we have

$$\sum_{k=0}^{l-1} \alpha'_{\sigma^*_{\beta,k}} \cdot \sum_{i=0}^{fs-1} \vartheta^{(a)}_i \cdot z^i = 0$$

which means that $\forall i \in [0, fs-1], \vartheta^{(a)}_i = 0$ and therefore the independence of $v^*_{j+1} \cdot \sum_{k=0}^{l-1} \alpha'_{\sigma^*_{\beta,k}} \cdot \alpha^{j+k+1}_{\sigma^*_{\beta,k}} \cdot z^k$ which concludes the proof.