Pattern Matching over 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 several application domains such as digital forensics, cyber threat intelligence, or genomic and medical data analysis. 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 new provably correct, secure, and relatively (compared to similar existing schemes) efficient public and private key based constructions that allow arbitrary pattern matching over encrypted data while 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 plain-texts). Compared to existing solutions, the constructions we propose has some interesting 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 on the size of the data to be inspected and is constant on the sizes of the analysis patterns. The conducted evaluations show that our constructions drastically improve the performance of the most efficient state of the art solution.

Keywords: Searchable encryption · Pattern Matching

1 Introduction

In several application domains such as deep-packet inspection and genomic data analysis, learning the presence of specific patterns as well as their positions in the data are essential. In the previous two use cases, pattern searches are often performed by entities that are not fully trusted by data owners. For instance, in the case of deep-packet inspection (DPI), a company that aims to outsource its network traces to a third party forensic scientist to find indictors of compromise might not be comfortable revealing the full contents of its traces to the forensic scientist. Similarly, in the case of genomic data analysis, a patient that wants to check whether its genome contains particular patterns representing a genetic predisposition to specific diseases might not be comfortable revealing the full contents of its genome to the laboratory that performs the analysis.

Existing solutions that may be used to overcome the previous problem rely mainly on searchable encryption based techniques [1–7, 21]. 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 [2–5, 21]); 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 two technically sound constructions: $\mathrm{S}^{\mathbf{4}}\mathrm{E}$ supporting pattern matching of adaptively chosen and variable (upper bounded) lengths patterns on secret key encrypted streams, and $\mathrm{AS}^{\mathbf{3}}\mathrm{E}$ supporting pattern matching of adaptively chosen and variable (upper bounded) lengths patterns on public key encrypted streams. Both $\mathrm{S}^{\mathbf{4}}\mathrm{E}$ and $\mathrm{AS}^{\mathbf{3}}\mathrm{E}$ ensure that (1) the third-party entity performing pattern matching operations will learn nothing about the searched patterns except their lengths, (2) the third-party entity that is going to perform pattern matching will learn nothing about the data to be analyzed except the presence or the absence of the set of unknown patterns (i.e., the third-party entity will not have access to patterns plaintexts), (3) the third-party entity will be able to perform pattern matching correctly over the data to be analyzed. From a practical point of view, our construction has some interesting 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. The two constructions we propose in this paper are – to our knowledge – the first constructions to provide all previously mentioned properties without using costly and complex cryptographic scheme such as fully homomorphic encryption. The evaluations conducted in this paper show that the two proposed constructions improve by up to four orders of magnitude 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 assumptions under which our schemes achieve provable security. The intuition behind the proposed constructions is presented in Section 4. Section 5 and 6 formalize our $\mathrm{S}^4\mathrm{E}$ and $\mathrm{AS}^3\mathrm{E}$ primitive and provide their security results. In Sections 7 and 8, we discuss the complexity of our construction and provide experimental results. Finally, section 9 concludes.

2 Related Work

One possible solution for pattern matching over encrypted traffic is to use techniques that allow evaluation of functions over encrypted data. Generic ap-

proaches 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, 21]. 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?=). The two constructions we propose in this paper offer better search flexibility as, even after the plaintext has been encrypted, they can allow arbitrarily chosen keywords to be searched 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 DPI 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. The two constructions we propose address the main drawbacks of these tokenization-based techniques since they allow for arbitrary trapdoors 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.

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 for 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 1GB of data). Moreover, the trapdoor generation technique used by the SEST leaks many information (such as, the number of different characters, the maximum number of occurrences of a character) about the pattern. 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 keyword. This makes the proposed technique quite inefficient when used for bit level matching. By contrast, for testing the presence of a pattern in encrypted data, our schema requires a constant number of pairings in the size of the pattern (see Section 7 for more details). This makes our construction more efficient when matching long keyword 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 the two approaches we propose in this paper 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, 21], induce higher complexity [9, 8, 10], require interactivity [11–13] or ensure weaker privacy level [5].

According to the Table 1, the two constructions we propose in this paper (S^4E and AS^3E) are the only primitives that simultaneously enable arbitrary trapdoors (with upper bounded keyword size), provides a correct keyword detection, and ensures the privacy of the used trapdoors.

In Table 1, (✔) is used to denote that a property is provided under specific conditions. AS^3E ensures trapdoor's privacy for patterns of high-min entropy (see Section 6 for more details). In addition, both S^4E and AS^3E support pattern matching of arbitrary but upper bounded lengths patterns. Fortunately, as we show in Section 7, in both S^4E and AS^3E, increasing the upper bound size of pattern affects only the size of trapdoor that will be generated for each pattern. The size of later increases linearly with the increase of the size of the former.

	Primitives					
	BlindBox	BlindIDS	PE/HVE	SEST	S^4E	AS^3E
Number of Trapdoors	$O(s \cdot q)$	$O(q)$	$O(n \cdot q)$	$O(q)$	$O(q)$	$O(q)$
Public Parameters size	$O(1)$	$O(1)$	$O(1)$	$O(1)$	$O(l_i)$	$O(1)$
keys size	$O(1)$	$O(1)$	$O(n)$	$O(n)$	$O(l_i)$	$O(l_i)$
Ciphertext size	$O(n \cdot L)$	$O(n \cdot L)$	$O(n)$	$O(n)$	$O(n)$	$O(n)$
Number of trapdoors	$O(q)$	$O(q)$	$O(n \cdot q)$	$O(q)$	$O(q)$	$O(q)$
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	$2 \cdot q \cdot n$ pairings
Arbitrary trapdoors	✗	✗	✓	✓	(✓)	(✓)
Trapdoor's privacy	✗	(✓)	✗	✗	✓	(✓)
Correctness (no false positives)	✗	✗	✓	✓	✓	✓

Table 1: Complexity and ensured security properties comparison between related work and our primitive. The scalars n, q, l_i, L, s denotes respectively the length of the traffic to encrypt, the number of pattern to be searched, the length of each pattern, the number of different lengths among the q patterns to be searched and the number of data encrypters. We used (✓) to denote that the property is provided under specific conditions.

The two construction we propose do not require very large public parameters or secret key or very large public keys as SEST and PE/HVE. Moreover, their search complexities is lower than SEST by a factor of l_i (the length of the pattern w_i to be searched), since it is constant in the size of the pattern to be searched. Therefore, the proposed constructions are an interesting middle way which almost provides the best of PE/HVE and SEST while ensuring the trapdoors privacy. Their only drawback compared to PE/HVE and SEST is the upper bounded size of patterns to be searched 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 Security Assumption

In this section, we describe the security assumptions that our two construction S^4E and AS^3E rely on.

Definition 1 (Bilinear Maps). *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 $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ 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*

As in [1], the security of the proposed constructions hold 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. Some of the security proofs of the proposed constructions, given in Appendix A, rely partially on showing that given a number of pattern trapdoors, the adversary will be unable to distinguish a valid trapdoor from a random element. As a result, the leakage can be bounded only by considering the adversary's query to the issuing oracle. Hence, either we considerably reduce the maximum length of the patterns to be searched (≤ 30), which allow to define a GDH instance providing all public parameters, the trapdoors for all possible patterns and the challenge elements, or we use an interactive variant of the GDH assumption offering flexibility to the simulator to allow the elements $g^{R^{(i)}(x_1,\cdots,x_c)}$, $\widetilde{g}^{S^{(i)}(x_1,\cdots,x_c)}$, and $e(g,\widetilde{g})^{T^{(i)}(x_1,\cdots,x_c)}$ of the GDH assumption [20] to queried to specific oracles.

So, we prove the security of the proposed constructions under an interactive assumption. That is, we use a slightly modified General Diffie-Hellman (GDH) problem assumption [20] 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 and the number 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 analysis pattern query is not obvious and may justify the use of such an interactive assumption.

Definition 2 (independence [20]). *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 3 (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 $\mathcal{R}$, (resp. $\mathcal{O}^{\mathcal{S}}$ 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. $\mathcal{O}^{S}$ and $\mathcal{O}^{T}$).*

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 $\{g^{R^{(i)}(x_1,\cdots,x_c)}\}_{i=1}^{r+k\cdot q_r}$, $\{\widetilde{g}^{S^{(i)}(x_1,\cdots,x_c)}\}_{i=1}^{s+k\cdot q_s}$, 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 (i) $U = g^{f(x_1,\cdots,x_c)}$ or U is random and (ii) $U' = \widetilde{g}^{f(x_1,\cdots,x_c)}$ 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 [20]. That is, as long as the challenge polynomial that we denote f is independent of $< R, S, T >$, an adversary cannot distinguish $g^{f(x_1,\cdots,x_c)}$ (resp. $\widetilde{g}^{f(x_1,\cdots,x_c)}$) from a random element of $\mathbb{G}_1$ (resp. $\mathbb{G}_2$). 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.

4 The intuition

The intuition behind the proposed constructions relies on two observations. First, the number of analysis patterns is often very small compared to the quantity of data that are going to be analyzed, e.g., in a deep packet inspection scenario, the number of patterns provided by the SNORT intrusion detection system is 3734 [22]. Second, the sizes of the detection patterns are also 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 data with alphabet Σ, the proposed constructions associate each element σ of Σ with a secret encoding $(\alpha'_\sigma, \alpha_\sigma)$. They fragment the sequence of symbols that represents the data $\mathcal{B}$ as described in the Figure 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 a pattern. In the proposed constructions, we require that $fs \geq 2 \cdot (p_{max} - 1)$. In the rest of the paper, we will use $\{x_i\}_{i=a}^{i=b}$ to denote the set of elements x_i, $i \in [a, b]$.

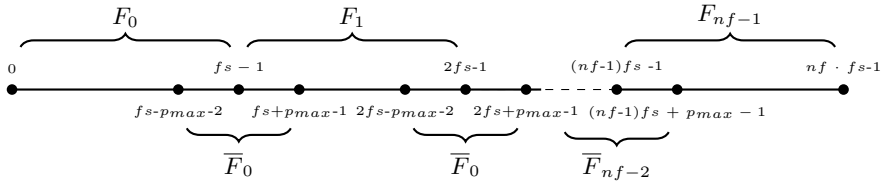


Fig. 1: Fragmentation approach

As illustrated by the Figure 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 $w = \sigma_{w,0} \cdot \sigma_{w,l-1}$ will be associated with a set of polynomials $\{V_i = v_i \sum_{k=0}^{l-1} \alpha'_{\sigma_{w,k}} \cdot \alpha_{\sigma_{w,k}}^{k+i} \cdot z^k\}_{i=0}^{i=fs-1}$ 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 of the previous elements will be used to check the presence of w at an index of the previously constructed fragments.

Meanwhile, the encryption of each symbol σ_i is the tuple $\mathcal{C}_i = \{C_i, C'_i, \overline{C}_i, \overline{C'}_i\}$ which 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 would be useless to the adversary.

5 S^{4}E Construction

In this section, we propose S^{4}E, a construction that supports pattern matching of adaptively chosen and variable (upper bounded) lengths patterns on secret key encrypted streams. Before formalizing S^{4}E, we present a use-case scenario on which S^{4}E can be useful.

5.1 Usage Scenario

To cope with new and sophisticated cybercrime threats, new threat intelligence platforms such as [19] are relying on the collaboration between different involved entities that include, on one side, companies, organizations, and individuals that are targeted by cyber attacks, and on the other, security editors that are in charge of defining and providing strategies for effectively detect and prevent cyber attacks. To be useful, such platforms should, on one hand, 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. On the other hand, 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.

The S^4E construction can be used to overcome the previous two limitations by building a platform that is (1) market compliant meaning that the third-party entity performing the pattern matching operations will learn nothing about the patterns that will be used by security editors for analyzing the shared traces (as proved by Theorem 4), and (2) privacy-friendly, signifying that the third-party performing the pattern matching operations will learn nothing about the shared traces except the presence or the absence of a set of unknown analysis patterns (as proved by Theorem 2).

5.2 Considered Architecture

The architecture considered for the S^4E construction involves three main parties: the data owner (DO) representing the entity that holds the data to be analyzed (e.g., the network traces in the case of DPI), the patterns provider (PP) representing the entity that supplies the patterns that will be searched, and the Service Provider (SP) are stakeholders that offer computation infrastructures that will be used for pattern matching on the data to be analyzed. SP is in charge of notifying the DO with the results of the pattern matching (i.e., the patterns that are present in the DO's data and their corresponding position in the DO's data) performed over the data provided by the DO.

5.3 Security Requirements and Considered Hypothesis

Before presenting the formal description of the S^4E construction, we first present the security properties that S^4E are supposed to provide.

PP is considered in S^4E as an *Honest-but-curious* entity. That is we expect PP to provide valid patterns that allow DO to effectively analyze its data. This a fairly reasonable assumption since a PP (e.g., a security editor in the case of DPI or a laboratory in the case of genomic data analysis) will not defile its reputation by issuing incorrect or misleading analysis patterns. Otherwise, this will result in many false positives, which may considerably degrade the quality of the analyses that will be provided to the DO. Nevertheless, we expect the

PP to be curious: it may try to derive information about the analyzed data by accessing the data analyzed by the SP or/and the analyses results returned by the SP to the DO. Hypothesis 5.1 and Requirement 5.1 summarize the previous considerations.

Hypothesis 5.1 *PP will perform queries that are only designed to analyze the outsourced cyber attak traces.*

Requirement 5.1 *PP 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 perform the the pattern matching operations honestly over the DO's data using the analysis patterns provided by the PP. However, we suppose that the SP may try to learn additional information about either or both the DO's outsourced network traces and the analysis patterns provided by the SE. In addition, we assume that the SP that may try to create values by analyzing other third-parties network traces using the set of patterns provided by the PP for the analysis of DO's network traces. We then consider the following hypothesis and security requirements.

Hypothesis 5.2 *SP will correctly perform the pattern matching operations required by the PP's analysis queries and will correctly report analysis results to the PP.*

Requirement 5.2 *SP should learn nothing about the analyzed traces except the indexes in which the analysis patterns fully match the (a part of) traces.*

Requirement 5.3 *SP should learn nothing about the analysis patterns (queries) performed by the PP except the indexes in which the patterns fully match (a part of) the traces.*

Requirement 5.4 *The patterns used by the PP 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 PP's patterns used to analyze its data. Moreover, we suppose that the SP and PP 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 5.3 *SP and PP will not collude to gain more information about the DO's outsourced traces.*

Finally, we require our construction to provide correct results. That is, (1) any part of the DO's data that matches one of the PP's patterns when not encrypted must be detected by S^4E (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 5.5 *The proposed construction should provide a correct pattern matching results.*

5.4 Syntax of S^{4}E

Our construction 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 between the DO and the PP. The **Test** algorithm is performed by the SP.

- **Setup**$(1^\lambda, fs, p_{max})$ is a probabilistic algorithm that takes an input a security parameter λ, the size of the fragment to be encrypted, and the maximum size of a pattern. It returns the public parameters $params$.
- **Keygen**$(params, \Sigma)$ is a probabilistic key generation algorithm that takes as input the public parameters $params$ and a finite set Σ representing the alphabet we are going to consider. It outputs a secret key K.
- **Encrypt**$(params, K, \mathcal{B})$ is a probabilistic algorithm that takes as input the public parameters $params$, a finite sequence (String) of elements $\mathcal{B}$ of Σ of size n and the secret key K. it returns a ciphertext C.
- **Issue**$(params, K, w)$ is a probabilistic algorithm that takes as input the public parameters $params$, the secret key K, and w – a finite sequence of elements of Σ, and returns a trapdoor td_w.
- **Test**$(params, C, td_w)$ is a deterministic algorithm that takes as input the public parameters $params$, a ciphertext C encrypting a sequence of m elements $\mathcal{B} = \sigma_0 \cdots \sigma_{m-1}$ of Σ, and the trapdoor td_W 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 $\mathcal{I} \subset \{0, m-l-1\}$ 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 S^{4}E 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.

5.5 S^{4}E's security requirements

As said in Section 5.3, there are mainly 4 security requirements that should be satisfied by our construction; Trace indistinguishability (Requirements 5.1 and 5.2), pattern indistinguishability (Requirement 5.3), pattern usefulness (Requirement 5.4), and the correctness property (Requirement 5.5).

In the following, we use the game-based security definition proposed in [1] for trace indistinguishability by adapting the standard notion of IND-CPA which requires that no adversary $\mathcal{A}$ (e.g., the PP 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 contains T_1 or T_2 as long as the trapdoors $\{td_{p_i}\}$ issued by $\mathcal{O}^{iss}$ do not allow trivial distinction of the traces T_1 and T_2. We note that we consider the quite standard *selective* security notion [17]. This notion requires the adversary to choose and commit $\mathcal{T}_0$ and $\mathcal{T}_1$ at the beginning of the experiment, before seeing $params$.

Definition 4 (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 $Exp^{S^4E\_T\_IND\_CPA}_{\mathcal{A},\beta}$:*

(1) Setup: $\mathcal{C}$ executes $Setup(1^\lambda, fs, p_{max})$ to generate params and the algorithm $Keygen(params, \Sigma)$ to generate the secret key K. Then it sends params to the adversary.
(2) Query: $\mathcal{A}$ can adaptively ask $\mathcal{O}^{iss}$ for the trapdoor td_{p_i} for any pattern $w_i = \sigma_{i,0} \cdots \sigma_{i,m-1}$ where $\sigma_{i,j} \in \Sigma$. We denote $\mathcal{W}$ 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 data streams $T_0 = \sigma^*_{0,0} \cdots \sigma^*_{0,m-1}$ and $T_1 = \sigma^*_{1,0} \cdots \sigma^*_{1,m-1}$ and sends them to $\mathcal{C}$.
(a) If $\exists w = \sigma_0 \cdots \sigma_l \in \mathcal{W}$, $k \in \{0,1\}$, and j such that:

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

(b) $\mathcal{C}$ chooses a random $\beta \in \{0,1\}$, creates $C = Encrypt(param, K, T_\beta)$, 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 $Adv^{Exp^{S^4E\_T\_IND\_CPA}_{\mathcal{A},\beta}}(\lambda) = |Pr[\beta = \beta'] - 1/2|$. S^4E is said to be trace indistinguishable if $Adv^{Exp^{S^4E\_T\_IND\_CPA}_{\mathcal{A},\beta}}(\lambda)$ is negligible.

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

We want to be able to evaluate the advantage of the SP for using the issued trapdoors to analyze other third-parties' data (i.e., data that are not provided and encrypted by the DO). Since encrypted data 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 data.

Definition 5 (Encrypted Data Forgery). *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 data. We consider the following $Exp^{S^4E\_ETF}_{\mathcal{A}}$ game:*

(1) Setup: $\mathcal{C}$ *executes* $Setup(1^\lambda, fs, p_{max})$ *to generate params and the algorithm* $Keygen(params, \Sigma)$ *to generate the secret key* K*. Then it sends params to the adversary.*

(2) Query:

- $\mathcal{A}$ *can adaptively ask* $\mathcal{O}^{iss}$ *for the trapdoor* td_{p_i} *for any 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* $C^T = Encrypt(params, K, T)$*. We denote* $\mathcal{T}$ *the set of datasets encrypted by the* $\mathcal{O}^R$*.*

(3) Forgery: The adversary chooses the dataset $T^* \notin \mathcal{T}$ *such that* T^* *contains* p_t $(p_t \in \mathcal{P})$ *at index* i *and forges the encrypted dataset* C^* *of the* T^* *.*

We define $\mathcal{A}$*'s advantage of winning the game* $Exp_{\mathcal{A}}^{S^4E\_ETF}$ *by* $Adv^{Exp_{\mathcal{A}}^{S^4E\_ETF}}(\lambda) = Pr[i \in Test(params, C^*, td_{p_t})]$*.* S^4E *is said to be encrypted data forgery secure if* $Adv^{Exp_{\mathcal{A}}^{S^4E\_ETF}}(\lambda)$ *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 6 (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* $Exp_{\mathcal{A}}^{S^4E\_P\_IND\_CPA}$*:*

(1) Setup: $\mathcal{C}$ *executes* $Setup(1^\lambda, fs, p_{max})$ *to generate params and the algorithm* $Keygen(params, \Sigma)$ *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* $\mathcal{O}^{iss}$ *for the trapdoor* td_{w_i} *for any 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* $Exp_{\mathcal{A},\beta}^{S^4E\_P\_IND\_CPA}$ *by* $Adv^{Exp_{\mathcal{A},\beta}^{S^4E\_P\_IND\_CPA}}(\lambda) = |Pr[\beta' = \beta] - 1/2|$*.* S^4E *is said to be pattern indistinguishable if* $Adv^{Exp_{\mathcal{A},\beta}^{S^4E\_P\_IND\_CPA}}(\lambda)$ *is negligible.*

Definition 7 (Correctness). *Given $\mathcal{B} = \sigma_0, \cdots \sigma_{m-1}$ and $p = \sigma_{p,0}, \cdots \sigma_{p,l_t-1}$ that represents respectively the data and the pattern to be searched. S^4E is consistent iff the following conditions hold:*

(i) $Pr[i \in Test(params, Encrypt(params, \mathcal{B}, K), Issue(params, p, K))] = 1$ *if $\mathcal{B}$ contains p at index i.*

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

Condition (i) of the previous definition ensures that the Test algorithm used in our construction produces no false negatives. Condition (ii) 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.

5.6 A trivial Protocol

A trivial attempt for defining a construction that ensures all of the security requirements we defined in Section 5.3 would consist of modifying the most efficient state of the art solution SEST [1] 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 PP, while the Test algorithm will be performed by the SP.

- **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. We denote by L the array that will be used to store random scalars that will be used to encode each symbol of the pattern w, and $\mathcal{I}$ is an array of sets representing the indices of symbols in w that are encoded using the same random scalar. Actually, a random scalars can be re-used as long as it has not been used to encode the same symbol. That is, $L[i]$ is the random scalar to use with the (imperatively distinct) symbols at indices $\mathcal{I}_i$ of w.

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 inherits the three main limitations of the SEST construction. 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).

5.7 The S^{4}E's Protocol

– **Setup**$(1^\lambda, fs, p_{max})$: 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**$(params, \Sigma)$: 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**$(params, \mathcal{B}, K)$: 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 $F_i = [i \cdot fs, (i+1) \cdot fs - 1]$ and $\overline{F}_j = [(j+1) \cdot fs - p_{max} - 2, (j+1) \cdot fs + p_{max} - 1]$. 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 the following algorithm.

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

Input: $K, params, w = \sigma_{w,0}, \cdots \sigma_{w,l-1}$
Output: $td_w = \{V_i, v_i\}_{i=0}^{fs-l-2}$
$td_w \leftarrow \emptyset$
foreach $i \in [0, fs-l-2]$ **do**
  $v_i \xleftarrow{\$} \mathbb{Z}_p$
  $V_i = v_i \cdot \sum_{k=0}^{l-1} \alpha'_{\sigma_{w,k}} \cdot \alpha_{\sigma_{w,k}}^{k+i} \cdot z^k$
  $td_w \leftarrow td_w \cup \{\widetilde{g}^{V_i}, \widetilde{g}^{v_i}\}$
end

Algorithm 3: Issue

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

Input: $C = \{C_i, \overline{C}_i, C'_i, \overline{C'}_i\}_{i=0}^{m-1}, td_w = \{V_i, v_i\}_{i=0}^{i=fs-l-2}$
Output: $\mathcal{I}$
$\mathcal{I} \leftarrow \emptyset$
foreach $i \in [0, m-1]$ **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 $e(\prod_{j=0}^{l-1} \overline{C}_{i+j}, \widetilde{g}^{v_{i_{\overline{F}_\epsilon}}}) = e(\overline{C'}_i, \widetilde{g}^{V_{i_{\overline{F}_\epsilon}}})$ **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 the set of patterns to be used but depends only on the size of data to be encrypted. In addition, our Issue and Test algorithms are used to search an arbitrary (upper bounded size) and unforgeable (without the knowledge of the secret key K) patterns. The sizes of those 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 a pattern).

5.8 S^{4}E's Security Results

In this section, we prove that the S^{4}E construction described in Section 5.7 provides the security requirements we described in Section 5.3. Proofs of the following theorems are given in Appendix **??**.

Theorem 1. *S^4E is correct.*

Theorem 2. *S^4E is trace indistinguishable under the i-GDH assumption.*

Theorem 3. *S^4E is encrypted data forgery secure under the i-GDH assumption.*

Theorem 4. *S^4E is pattern indistinguishable under the i-GDH assumption.*

6 AS^{3}E Construction

The S^{4}E construction, introduced in Section 5, allows for pattern matching of upper bounded length keywords on symmetrically encrypted stream. In this section we show that the data fragmentation approach we propose in Section 4 can be used to build AS^{3}E: a pattern matching of upper bounded length

keywords on asymmetrically encrypted stream. In particular, we show in Section 7 that considering the same system and threat model as SEST [1], AS^{3}E is far more practical than SEST as it reduces (1) considerably the size of public keys and (2) slightly the search complexity while increasing the size of ciphertext only by a factor of 2.

6.1 Considered Architecture

The construction we propose in this paper involves four roles: Pattern Provider, Service Provider, sender, and receiver. Pattern provider (PP) and Service Provider (SP) are the same two entities we used in the S^{4}E construction. That is, (PP) the entity that supplies the patterns that will be searched, and the Service Provider (SP) are stakeholders that offer computation infrastructures that will be used for pattern matching on the data to be analyzed. The role sender is used to represents the entities that are going to generate the data that is going to be analyzed (e.g., a website that provides web contents). The role receiver represents the entities that will receive and process the traffic sent by the sender. The receiver and the sender roles are interchangeable. That is, within the same secure network connection session, each end-point may play both the sender and receiver roles. In this case, both the sender and receiver roles may be malicious. In our solution, we require that the sender and the receiver will not collaborate together, otherwise, they could use a secure channel that is out of reach for the SP. This scenario should not be considered as a limitation of our construction since, such scenario will get undetected even in the context of a plaintext traffic.

6.2 Security Requirements and Hypothesis

We consider the same hypothesis for the two entities PP and SP as in the construction S^{4}E. That is, PP ans SP are considered to be *honest-but-curious* entity. Specifically, PP is supposed to provide valid patterns that allow SP to effectively analyze the data generated by the sender while SP is supposed to perform correctly the matching between the patterns provided by PP and the sender's data. Nevertheless, we expect the PP and SP to be curious as the former may try to learn information about the sender's data and the latter may try to get additional information about both the the patterns provided by PP and the sender's data. In addition, we suppose that the SSP and SE will not collude to learn more information about the traffic since, otherwise, they could easily mount a dictionary attack. 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.

Moreover, we require the receiver to be honest. That is, he will correctly follow the protocol used in our construction without disclosing the patterns that are provided by PP.

Finally, as in S^{4}E, the detection provided by AS^{3}E should be correct in a way that (1) any traffic that matches a least one of the analysis patterns provided by PP when not encrypted must be detected as malicious traffic by

our construction, and (2) the probability that our construction returns a false positive for any traffic that does not match any of the PP's analysis patterns when not encrypted is negligible.

6.3 Syntax

Similarly to the S^{4}E construction, we used five algorithms to define our construction: Setup, Keygen, Encrypt, Issue, and Test. The algorithms Setup, Keygen, and Issue are performed using the entity playing receiver role. The Encrypt algorithm is performed by the sender while the Test algorithm is performed by the SP.

- Setup(1^λ, fs, p_{max}) is a probabilistic algorithm that takes as input a security parameter λ, the size of the partition (fragment) of the traffic to be encrypted fs, and the maximum size of a pattern p_{max}. It returns the public parameters $params$. The parameter $params$ is an implicit input to all other algorithms.
- Keygen(Σ) is a probabilistic algorithm that takes as input a finite set of symbols Σ representing the alphabet (e.g., bit values, byte values, etc.) used to represent the network traffic to be analyzed. It returns the pair of keys (sk, pk) where sk is private and known only to the receiver and pk is pubic.
- Encrypt($\mathcal{T}, pk$) is a probabilistic algorithm that takes as input the traffic to be encrypted along with the public key pk and returns a ciphertext $\mathcal{C}$.
- Issue(p, sk) is a probabilistic algorithm that takes as input the receiver's private key sk and a pattern p of length l ($l \leq p_{max}$) and returns a trapdoor td_p.
- Test($\mathcal{C}, td_p$) is a deterministic algorithm that takes as input a ciphertext $\mathcal{C}$ encrypting a traffic $\mathcal{T}$ along with a trapdoor td_p for a pattern p and returns the set of (potentially empty) indexes at which the pattern p occur in $\mathcal{T}$.

Similarly to our S^{4}E construction, we omit the decryption algorithm in the previous description since we focus mainly on providing arbitrary universal pattern matching over encrypted traffic. The decryption functionality can be easily added by encrypting the data stream $\mathcal{T}$ under a conventional encryption scheme, or by issuing a trapdoor for each character in the alphabet Σ and use the Test algorithm to decrypt the encrypted traffic.

6.4 Security Model

For the AS^{3}E construction, the are mainly three security requirements that should be satisfied: the traffic indistinguishability, the pattern indistinguishability, and the correct detection requirements. We note that, similarly to our S^{4}E construction, we consider the *selective* security notion [17].

The following definition states that it is not feasible for the SP o learn any information about the content of the traffic more than whether it is safe or malicious.

Definition 8 (Data 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 $Exp^{AS^3E\_T\_IND\_CPA}_{\mathcal{A},\beta}$:*

(1) Setup: $\mathcal{C}$ executes $Setup(1^\lambda, fs, p_{max})$ to generate params and the algorithm $Keygen(params, \Sigma)$ to generate the secret key sk and the public key pk. Then it sends params and pk to the adversary.
(2) Query: $\mathcal{A}$ can adaptively ask $\mathcal{O}^{iss}$ for the 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:

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

(b) $\mathcal{C}$ chooses a random $\beta \in \{0,1\}$, creates $C = Encrypt(param, pk, T_\beta)$, 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 $Adv^{Exp^{AS^3E\_T\_IND\_CPA}_{\mathcal{A},\beta}}(\lambda) = |Pr[\beta = \beta'] - 1/2|$. S^4E is said to be data indistinguishable if $Adv^{Exp^{AS^3E\_T\_IND\_CPA}_{\mathcal{A},\beta}}(\lambda)$ is negligible.

The pattern indistinguishability property informally requires that it is not feasible for an adversary (e.g., the SP) to learn any information about the detection patterns. Since our construction is a public-key based schema, we need to take into consideration the fact that an adversary can create any traffic of its choice using the public key *pk*. In this case, an adversary can mount a brute force attack on detection patterns by adaptively creating a lot of traffic to understand the logic behind the detection patterns. However, a pattern matching-based solution over plaintext or encrypted traffic cannot resist such a brute force attack, and therefore, it should not be considered in AS^{3}E security model. Hence, the pattern indistinguishability property mainly requires that the adversary $\mathcal{A}$ will not learn more information than what is provided as output to the Test algorithm. Formally, we use the high-min entropy property [18] which informally states that $\mathcal{A}$ cannot obtain the searched patterns "by chance".

Definition 9 (min-entropy). *Given a set of detection patterns $\mathcal{P}$,and a random bit $\beta \in \{0,1\}$. A probabilistic adversary $\mathcal{A} = (\mathcal{A}_f, \mathcal{A}_g)$ has min-entropy μ if*

$$\forall \lambda \in \mathbb{N}, \forall p \in \mathcal{P}, \forall \beta : Pr[p' \leftarrow \mathcal{A}(\lambda, \beta) : p = p'] \leq 2^{-\mu(\lambda)}$$

$\mathcal{A}$ is said to have high-min entropy if it has min-entropy μ with $\mu(\lambda) \in \omega(\log(\lambda))$.

In the experiment $Exp^{AS^3E\_p\_sIND}_{\mathcal{A}=(\mathcal{A}_f,\mathcal{A}_g),\beta}$ (Definition 10), we define the security notion $AS^3E\_P\_sIND$ for an adversary $\mathcal{A} = (\mathcal{A}_f, \mathcal{A}_g)$ ($\mathcal{A}_f$ and $\mathcal{A}_g$ are non colluding entities, as in e.g., [18, 7]) with high-min entropy, that can create any traffic of its choice.

Definition 10 (Pattern indistinguishability). *Let λ be the security parameter, Σ be the alphabet to be used, $\mathcal{A} = \mathcal{A} = (\mathcal{A}_f, \mathcal{A}_g)$ be the adversary and $\mathcal{C}$ be the challenger. We consider the following game that we denote $Exp^{AS^3E\_p\_sIND}_{\mathcal{A}=(\mathcal{A}_f,\mathcal{A}_g),\beta}$:*

(1) Setup: $\mathcal{C}$ executes $Setup(1^\lambda, fs, p_{max})$ to generate params and the algorithm $Keygen(params, \Sigma)$ to generate the secret key sk and the public key pk. Then it sends params and pk to the adversary $\mathcal{A}$.
(2) Query: $\mathcal{A}$ can adaptively ask $\mathcal{O}^{iss}$ for the trapdoor td_{w_i} for any pattern $p_i = \sigma_{i,1} \cdots \sigma_{i,l}$ 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.
(3) Challenge: Once $\mathcal{A}$ decides that Phase (2) is over, $\mathcal{A}_f$ chooses two patterns $p_0^* = \sigma_{0,0}^* \cdots \sigma_{0,l}^*$ and $p_1^* = \sigma_{1,0}^* \cdots \sigma_{1,l}^*$ such that $p_0^*, p_1^* \notin \mathcal{P}$ and sends them to $\mathcal{C}$. If $\exists T \in \mathcal{T}$ such that $p_0^* \in T$ or $p_1^* \in T$ then return 0. Otherwise, $\mathcal{C}$ chooses a random $\beta \in \{0,1\}$, creates $td_{p_\beta^*}$, and sends it to $\mathcal{A}_g$.
(4) Guess: $\mathcal{A}_g$ outputs the guess β'.
(5) Return ($\beta = \beta'$).

We define $\mathcal{A}$'s advantage by $Adv^{Exp^{AS^3E\_p\_sIND}_{\mathcal{A}=(\mathcal{A}_f,\mathcal{A}_g),\beta}}(\lambda) = |Pr[\beta = \beta'] - 1/2|$. S^4E is said to be trace indistinguishable if for any probabilistic polynomial-time $\mathcal{A} = (\mathcal{A}_f, \mathcal{A}_g)$ having high-min entropy $Adv^{Exp^{AS^3E\_p\_sIND}_{\mathcal{A}=(\mathcal{A}_f,\mathcal{A}_g),\beta}}(\lambda)$ is negligible.

Finally, the detection correctness property is formally defined in the following Definition.

Definition 11 (Correctness). *Given $\mathcal{T} = \sigma_0, \cdots \sigma_{m-1}$ and $p = \sigma_{p,0}, \cdots \sigma_{p,l_t-1}$ that represent respectively a traffic and a detection pattern w. Our construction is correct iff the following conditions hold:*

(i) $Pr[i \in Test(Encrypt(\mathcal{T}, pk), Issue(w, sk))] = 1$ if $\mathcal{T}$ contains w at index i.
(ii) $Pr[i \in Test(Encrypt(\mathcal{T}, pk), Issue(w, sk))]$ is negligible if $\mathcal{T}$ does not contain w at index i.

Condition (i) of the previous definition ensures that the Test algorithm used in our construction produces no false negatives. Condition (ii) 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.

6.5 The protocol

- **Setup**$(1^\lambda, fs, p_{max})$: 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 sets the public key $pk \leftarrow \{g^{z^i}, g^{\alpha'_\sigma \cdot (\alpha_\sigma \cdot z)^i}\}_{i=0,\sigma\in\Sigma}^{i=fs-1}$ and the private key $sk \leftarrow \{\alpha_\sigma, \alpha'_\sigma, z\}_{\sigma \in \Sigma}$. It returns (sk,pk).

– **Encrypt**$(\mathcal{B}, pk)$: This algorithm fragments $\mathcal{B} = \sigma_1, \cdots \sigma_m$ into $\{F_i, \overline{F}_j\}_{i=0,j=0}^{i=nf-1,j=nf-2}$ where $F_i = [i \cdot fs + 1, (i+1) \cdot fs]$ and $\overline{F}_j = [(j+1) \cdot fs - p_{max} - 1, (j+1) \cdot fs + p_{max}]$. 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 $C = \{C_i, \overline{C}_i, C'_i, \overline{C'}_i\}_{i=1}^{m}$ as described in the following algorithm.

Input: $pk, params, \mathcal{B} = \sigma_1, \cdots \sigma_m,$
$\{F_i, a_i, \overline{F}_j, \overline{a}_j\}_{i=0,j=0}^{i=nf-1,j=nf-2}$
Output: $C = \{C_i, \overline{C}_i, C'_i, \overline{C'}_i\}_{i=1}^{m}$
$C \leftarrow \emptyset$
foreach $i \in [1, m]$ **do**
  $\epsilon \leftarrow i/fs$ #find the fragment F_ϵ to which i belongs
  $C_i \leftarrow g^{a_\epsilon \cdot \alpha'_{\sigma_i} \cdot (\alpha_{\sigma_i} \cdot z)^{i_{F_\epsilon}}}$, $C'_i \leftarrow g^{a_\epsilon \cdot z^{i_{F_\epsilon}}}$ # $g^{\alpha'_{\sigma_i} \cdot (\alpha_{\sigma_i} \cdot z)^{i_{F_\epsilon}}}$ and $g^{z^{i_{F_\epsilon}}}$ are retrived from pk
  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 5: Encrypt

– **Issue**$(params, w, sk)$ issues a trapdoor td_w for the sequence of symbols $w = \sigma_{w,0}, \cdots, \sigma_{w,l-1}$ of length $l < p_{max}$ as described in the following algorithm.

Input: $sk, params, w = \sigma_{w,0}, \cdots \sigma_{w,l-1}$
Output: $td_w = \{V_i, v_i\}_{i=0}^{fs-l-2}$
$td_w \leftarrow \emptyset$
foreach $i \in [0, fs - l - 2]$ **do**
  $v_i \xleftarrow{\$} \mathbb{Z}_p$
  $V_i = v_i \cdot \sum_{k=0}^{l-1} \alpha'_{\sigma_{w,k}} \cdot \alpha_{\sigma_{w,k}}^{k+i} \cdot z^k$
  $td_w \leftarrow td_w \cup \{\widetilde{g}^{V_i}, \widetilde{g}^{v_i}\}$
end

Algorithm 6: Issue

– **Test**(C, td_w) tests whether the encrypted traces C contains the sequence of symbols w. It returns the set $\mathcal{I}$ of indexes i in which w exists in C. The Test algorithm is the same as described for the S^4E construction (Algorithm 4).

6.6 AS^{3}E Security Results

In this section, we prove that the AS^{3}E construction described in Section 6.5 provides the security requirements we described in Section 6.4. Proofs of the following theorems are given in Appendix **??**.

Theorem 5. *AS^{3}E is correct.*

Theorem 6. *AS^{3}E is trace indistinguishable under the i-GDH assumption.*

Theorem 7. *AS^{3}E is pattern-indistinguishable for patterns of high min-entropy under the i-GDH assumption.*

7 The complexity

We evaluate the practicability of S^{4}E and AS^{3}E regarding several properties: the sizes of the public parameters for S^{4}E, public keys for AS^{3}E, 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. Note that, S^{4}E and AS^{3}E share the same sizes for the ciphertext, the trapdoors, the upper bound size of patterns, and the same complexities for trapdoor generation, encryption, and search operations.

The size of the public parameters used in S^{4}E: The public parameters used in the S^{4}E construction contains fs elements of $\mathbb{G}_1$ which represents $32 \times fs$ bytes using Barreto-Naehrig (BN) [16].

The size of the public keys used in AS^{3}E: The public keys used in the S^{4}E construction contains $2 \times fs$ elements of $\mathbb{G}_1$ which represents $64 \times fs$ bytes using Barreto-Naehrig (BN) [16]. We underline that the size of the required public key is independent of the size of the data to be analyzed n and depends only on fs (fs $\ll$ n). Hence, compared to the most efficient state of the art solution SEST, AS^{3}E reduces considerably the size of the required public key. To illustrate, if we suppose that 1G of data is to be analyzed using a set of patterns, each composed of at most 10000 bytes, SEST requires a public key of size $32 \times (1 + 256) \times 10^9$ bytes $\simeq$ 8000 GB while AS^{3}E requires a public keys of size $\simeq$ 1.2 MB..

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 $\mathbb{G}_2$.

The upper bound size of patterns: The upper bound size p_{max} of the patterns that can be searched by S^4E and AS^3E depends on the size of the fragment fs used in the data fragmentation method detailed in section 4 ($p\_max = fs/2 - 1$). Increasing p_max will increase linearly the trapdoor's sizes and generation complexity. However, it will not affect any of the other properties of S^4E and AS^3E.

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 using S^4E 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 $\mathbb{G}_1$. As for AS^3E, encrypting a sequence of n symbols requires $2 \times n$ exponentiations in $\mathbb{G}_1$.

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 7. 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 [23] and that our Test procedure (Algorithm 7) is highly parallelizable.

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 7: Optimized pattern search Algorithm (Optimization instructions are highlighted)

8 Empirical Evaluation

In this section, we experimentally evaluate the performance of S^4E and AS^3E [3]. We implement the two constructions using the RELIC cryptographic library [23] over the 254-bits Barreto-Naehrig curve. 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 in

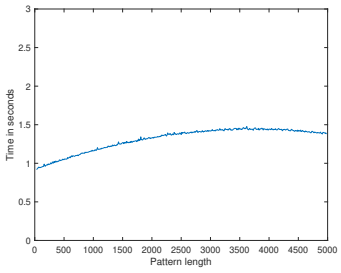
[3] We note that the goals of this section is to (1) provide a more concrete estimations of the different operations used by S^4E and AS^3E and (2) show that S^4E and AS^3E are more practical than SEST. Particularly, we do not claim that S^4E and AS^3E are practical enough to perform pattern matching on very large data streams.

both S^4E and AS^3E, the encryption and the trapdoor generation algorithms are to be performed by entities (data owners in case of S^4E or data sender in case of AS^3E) 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 7) 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.

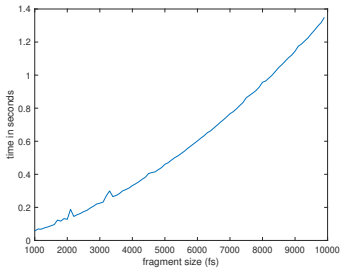
In our empirical evaluation, we aim to quantify the following characteristics of our constructions:

- 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 data stream 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 7) 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 a pattern w as a function of its length (Fig. 2 (a)) as well as the size fs of data a data fragment (Fig. 2 (b)). According to our experiments, issuing a trapdoor for a pattern of 5000 symbols take 1.4 second. In addition, the sizes of the generated trapdoors are relatively small (256 KB for a pattern of 4000 symbols).



(a) as a function of the number of symbols in the pattern to be searched



(b) as a function of fs

Fig. 2: Trapdoor generation time

Encryption time. According to Section 7, 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. Table 2 reports the time needed to encrypt a data stream $\mathcal{T}$ fragmented in chunks, each containing 1000 bits ($fs = 1000$ and $\Sigma = \{0, 1\}$), as a function of the data stream length n.[4].

data length (bytes)	**Time (seconds)**
1000	0.031
3000	0.097
5000	0.158
10000	0.371
30000	1.01
100000	3.0355

Table 2: Encrypting time as a function of the size of the data to be encrypted n

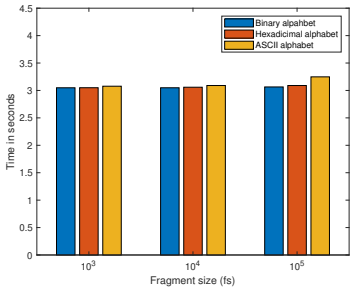


Fig. 3: Time required for encrypting 10^6 symbols as a function of the fragmentation size fs and the type of the considered alphabet Σ

As we noted in Section 4, the fragmentation size fs and the considered alphabets are important parameters in our construction. The former 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 latter parameter determines the type of search that can be performed by our construction. In Fig. 3, we compute the time required for the encryption of a data stream composed of 10^6 symbols 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. 3, the time required for encrypting a dataset composed of 10^6 symbols increases only by a factor of 0.02 (from 3,04 to 3,2 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 7, the complexity of the search operation (Algorithm 7) depends mainly on the number of encrypted symbols n that compose the data to be analyzed. Fig. 4 describes the time required for searching a pattern as a function of the number of encrypted symbols in the data to be analyzed.

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

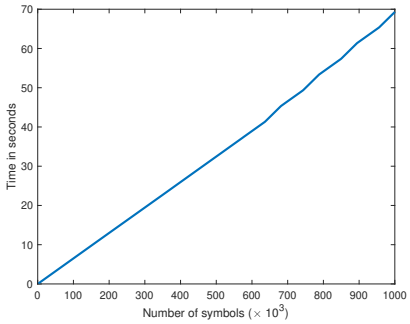
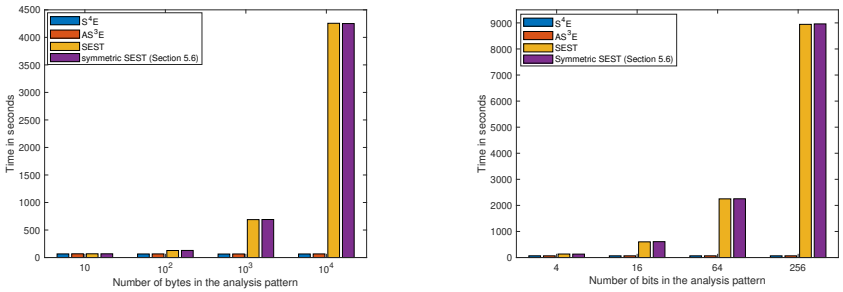


Fig. 4: Time required for searching a pattern as a function of the number of encrypted symbols in the data to be analyzed[3]

The conducted evaluations show that the average search throughput of our construction is 139078 symbols per second with a multi-threaded implementation [5]. Thus, if an ASCII (resp. binary) alphabet is considered, the search throughput is 139 KB (resp. Kb) per second.



(a) as a function of the number of symbols in the pattern to be searched

(b) as a function of fs

Fig. 5: Timing comparison for testing the presence of a pattern in a string of 10^6 symbols as a function of the pattern size

Fig. 5 (a) (resp. Fig. 5) compares the time needed for both our and the SEST (both its asymmetric [1] and symmetric (Section 5.6) variants) constructions to test the presence of a pattern of bytes (resp. of bits) 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 because that our Test algorithm (Algorithm 7) is constant on the size and on the content of the searched pattern which is not the case for SEST.

[5] search time would be roughly 96 times slower with a single-threaded execution.

9 Conclusion

In this work, we introduced two new provably correct and secure constructions S^4E and AS^3E. S^4E (resp. AS^3E) supporting pattern matching of adaptively chosen and variable (upper bounded) lengths patterns on secret key (resp. public key) encrypted streams. The proposed constructions has several interesting 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 constructions can be useful for other application scenarios such as subtrees search and searching of structured data.

To prove the security of the two proposed schemes, we used a slightly modified GDH assumption where the adversary is allowed to choose on which input to play the GDH instance. This relatively minor modification of the GDH assumption allow to define constructions that offer an interesting compromise between the secure and quite costly solutions and the fast and unsecure solution where the data has to be decrypted by the third-party entity that perform the pattern matching.

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

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.
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. 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.
18. 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.
19. MISP - Open Source Threat Intelligence Platform & Open Standards For Threat Information Sharing, https://www.misp-project.org/, 23 12 2011.
20. Xavier Boyen. The uber-assumption family. In *International Conference on Pairing-Based Cryptography*, pages 39–56. Springer, 2008.
21. Dawn Xiaoding Song, David Wagner, and Adrian Perrig. Practical techniques for searches on encrypted data. In *Proceeding 2000 IEEE Symposium on Security and Privacy. S&P 2000*, pages 44–55. IEEE, 2000.
22. Snort Rules. https://www.snort.org/, Accessed: 2019-08-35.
23. D. F. Aranha and C. P. L. Gouvêa. RELIC is an Efficient LIbrary for Cryptography. https://github.com/relic-toolkit/relic.

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** $i \in F_\epsilon \cap \overline{F}_\epsilon$

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

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

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

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:

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

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 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 $Exp_{\mathcal{A},\beta}^{T\_IND\_CPA}$ 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 $Exp_{\mathcal{A},\beta}^{T\_IND\_CPA}$ 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 $Exp_{\mathcal{A},\beta}^{T\_IND\_CPA}$ (Definition 4). 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 a random element 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 $Adv^{Exp^{T\_IND\_CPA}_{\mathcal{A},\beta}}(\lambda)$ can be bounded as following:

$$Adv^{Exp^{T\_IND\_CPA}_{\mathcal{A},\beta}}(\lambda) \leq \sum_{j=1}^{n-1} |G_j^{(1)}(\lambda) - G_{j+1}^{(1)}(\lambda)| + \sum_{j=1}^{n-1} |G_{j+1}^{(0)}(\lambda) - G_j^{(0)}(\lambda)|$$

Therefore, in order to show that $Adv^{Exp^{T\_IND\_CPA}_{\mathcal{A},\beta}}(\lambda)$ 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 $Exp^{T\_IND\_CPA}_{\mathcal{A},\beta}$ (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 $\overline{C'}_i = g^{\overline{a}_{\epsilon-1} z^{n+i_{\overline{F}_{\epsilon-1}} - i^*_{\overline{F}_{\epsilon-1}}}}$
- $\forall i \in \mathcal{I}^{(j)} : C_i, \overline{C}_i \xleftarrow{\$} \mathbb{G}_1$
- $\forall i \notin \mathcal{I}^{(j+1)}$ 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 $\overline{C'}_i = g^{\overline{a}_\epsilon z^{n+i_{\overline{F}_\epsilon} - i^*_{\overline{F}_\epsilon}}}$
- $\forall i \in \mathcal{I}^{(j)} : C_i, \overline{C}_i \xleftarrow{\$} \mathbb{G}_1$
- $\forall i \notin \mathcal{I}^{(j+1)}$ 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 $\overline{C'}_i = \emptyset$
- $\forall i \in \mathcal{I}^{(j)} : C_i \xleftarrow{R} \mathbb{G}_1$ 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 $Exp_{\mathcal{A},\beta}^{T\_IND\_CPA}$ 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.

Since $\alpha'_{\sigma_{i^*}} = x'_{h(\sigma_{i^*})} = x'_0$ and $\alpha_{\sigma_{i^*}} = x_{h(\sigma_{i^*})} = x_0$ we will replace $a \cdot x'_0 \cdot x_0^{i^*} \cdot z^n$ by $a \cdot \alpha'_{\sigma_{i^*}} \cdot \alpha^{i^*}_{\sigma_{i^*}} \cdot z^n$, for all $a \in \{a_{k^*}, \overline{a}_{k^*}, \overline{a}_{k^*+1}\}$.

Each pattern $p_t = \sigma_{t,0}, \cdots, \sigma_{t,l_t-1}$ submitted to $\mathcal{O}^S$ will add the polynomials $\sum_{s=0}^{fs-l_t-2} v_{t,s} \cdot \sum_{k=0}^{l_t-1} \alpha'_{\sigma_{t,k}} (\alpha_{\sigma_{t,k}} \cdot z)^{k+s}$ and $\sum_{s=0}^{fs-l_t-2} v_{t,s}$ to S. In addition, a query to the oracle $\mathcal{O}^R$ will add $\forall i \in [0, m-1] \backslash \{i^*\}$

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

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 $j \in [max(0, i^*+l_t-n), min(i^*, l_t-1)]$ form $\sigma^*_{\beta,i^*-j} \cdots \sigma^*_{\beta,i^*-j+l_t-1}$, 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

$$\left(a \cdot \alpha'_{\sigma^*_{i^*}} \cdot \alpha^{i^*}_{\sigma^*_{i^*}} \cdot z^n\right)\left(\sum_j u^a_j \cdot S^{(j)}\right) = \sum_{i,j} u^b_{i,j} \cdot R^{(i)} \cdot S^{(j)} + \sum_k u^{(c)}_k T^{(t)} \tag{2}$$

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 1). Thus, the elements of R that are not multiple of a cannot be part of Equation (2). In addition, the last sum of Equation (2) can be removed since T is empty. So, let $\{\vartheta^{(a)}_{i,t,s}, \vartheta^{(b)}_{i,t,s}, \vartheta^{(c)}_j, \vartheta^{(d)}_{j,t,s}, \vartheta^{(e)}_{j,t,s}, \vartheta^{(f)}_i\}_{i=0,j=0,t=1,s=0}^{i=\delta-1,j=2n-1,t=q,s=fs-1}$ be constants such that

$$\begin{aligned}
& a \cdot \alpha'_{\sigma^*_{i^*}} \cdot \alpha^{i^*}_{\sigma^*_{i^*}} \cdot z^n \Big(\sum_{t=1}^{q} \sum_{s=0}^{fs-l_t-2} (\vartheta^{(a)}_{i^*,t,s} \cdot V_{t,s} + \vartheta^{(b)}_{i^*,t,s} \cdot v_{t,s})\Big) \\
&= \sum_{j=0}^{2n-1} a \cdot \vartheta^{(c)}_j \cdot z^j \Big(\sum_{t=1}^{q} \sum_{s=0}^{fs-l_t-2} (\vartheta^{(d)}_{j,t,s} \cdot V_{t,s} + \vartheta^{(e)}_{j,t,s} \cdot v_{t,s})\Big) \\
&+ \Big(\sum_{i=0,i\neq i^*}^{\delta-1} a \cdot \vartheta^{(f)}_i \cdot \alpha'_{\sigma^*_i} \cdot \alpha^i_{\sigma^*_i} \cdot z^{n-i^*+i}\Big) \cdot \\
&\qquad \Big(\sum_{t=1}^{q} \sum_{s=0}^{fs-l_t-2} (\vartheta^{(a)}_{i,t,s} \cdot V_{t,s} + \vartheta^{(b)}_{i,t,s} \cdot v_{t,s})\Big)
\end{aligned}$$

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 $\vartheta^{(a)}_{i^*,t,s} = \vartheta^{(b)}_{i^*,t,s} = 0$ for any $s \in [0, fs-l_t-2]$ 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. $\sum_{j=0}^{2n-1} \vartheta^{(c)}_j \cdot z^j (\sum_{t=1}^{q} \sum_{s=0}^{fs-l_t-2} \vartheta^{(e)}_{j,t,s} \cdot v_{t,s}) = 0$
2. $\alpha'_{\sigma^*_{i^*}} \cdot \alpha^{i^*}_{\sigma^*_{i^*}} \cdot z^n (\sum_{t=1}^{q} \sum_{s=0}^{fs-l_t-2} \vartheta^{(b)}_{i^*,t,s} \cdot v_{t,s}) = \sum_{j=0}^{2n-1} \vartheta^{(c)}_j \cdot z^j (\sum_{t=1}^{q} \sum_{s=0}^{fs-l_t-2} \vartheta^{(d)}_{j,t,s} \cdot V_{t,s}) + (\sum_{i=0,i\neq i^*}^{\delta-1} \vartheta^{(f)}_i \cdot \alpha'_{\sigma^*_i} \cdot \alpha^i_{\sigma^*_i} \cdot z^{n-i^*+i})(\sum_{t=1}^{q} \sum_{s=0}^{fs-l_t-2} \vartheta^{(b)}_{i,t,s} \cdot v_{t,s})$

3. $\alpha'_{\sigma^*_{i^*}} \cdot \alpha^{i^*}_{\sigma^*_{i^*}} \cdot z^n (\sum_{t=1}^{q} \sum_{s=0}^{fs-l_t-2} \vartheta^{(a)}_{i^*,t,s} \cdot V_{t,s}) = (\sum_{i=0, i\neq i^*}^{\delta-1} \vartheta^{(f)_i} \cdot \alpha'_{\sigma^*_i} \cdot \alpha^{i}_{\sigma^*_i} \cdot z^{n-i^*+i})(\sum_{t=1}^{q} \sum_{s=0}^{fs-l_t-2} \vartheta^{(a)}_{i,t,s} \cdot V_{t,s})$

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

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

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

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

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 $\forall s \in [0, fs-l_t-2]$. It then only remains to prove that $\forall t \in [1,q], \forall s \in [0, fs-l_t-2] : \vartheta^{(b)}_{i^*,t,s} = 0$.

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

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

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:

$$\sum_{j=0}^{2n+l_t-2} z^j \sum_{k=0}^{l_t-1} \vartheta^{(c)}_{j-k} \cdot \vartheta^{(d)}_{j-k,t,s} \cdot \alpha'_{\sigma_{t,k}} \cdot \alpha^{s+k}_{\sigma_{t,k}} = -\sum_{i=0}^{\delta-1} \vartheta^{(f)}_i \cdot \alpha'_{\sigma^*_i} \cdot \alpha^{i}_{\sigma^*_i} \cdot z^{n-i^*+i} \cdot \vartheta^{(b)}_{i,t,s} \quad (3)$$

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 is 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 (3). 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 (3). 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 $td_{w_t} = \{V_j, v_j\}_{j=0}^{j=fs-l-2}$. 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 6, 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 **??**, 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 **??**, 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 $j \in [0, fs-1], \beta \in \{0,1\}$, $|Adv^{G_j^{(\beta)}}(\lambda) - Adv^{G_{j+1}^{(\beta)}}(\lambda)|$ is negligible under the i-GDH assumption where $f = v_{j+1}^* \cdot \sum_{k=0}^{l-1} \alpha'_{\sigma^*_{\beta,k}} \cdot \alpha^{j+k+1}_{\sigma^*_{\beta,k}} \cdot z^k$, $R = \{v_{t,s} \cdot \sum_{k=0}^{l-1} \alpha'_{\sigma_{t,k}} \cdot \alpha^{s+k}_{\sigma_{t,k}} \cdot z^k, v_{t,s}, v_s^* \cdot \sum_{k=0}^{l-1} \alpha'_{\sigma^*_k} \cdot \alpha^{s+k}_{\sigma^*_k} \cdot z^k, v_s^*\}_{t=1,s=0,s'=0,s'\neq j+1}^{t=q,s'=s=fs-1}$, $S = \{z^i\}_{i=0}^{i=fs-1}$, and $T = \emptyset$.*

Proof. According to the standard hybrid argument strategy we described before, in each G_{j+1}^{β}, to answer $\mathcal{A}$'s challenge the simulator uses the oracle $\mathcal{O}^S$ to get a valid trapdoor $td_{w^*_\beta} = \{\widetilde{g}^{V_s^*}, \widetilde{g}^{v_s^*}\}_{s=0}^{fs-l-2}$ 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 $G_j^{(\beta)}$. Otherwise, U is random and the simulator is playing the $G_{j+1}^{(\beta)}$. Then if $\mathcal{A}$ is able to distinguish $G_j^{(\beta)}$ and $G_{j+1}^{(\beta)}$ he/she will be able to win $Exp^{P\_IND\_CPA}_{\mathcal{A},\beta}$ with non negligible advantage. According to Definition 3, 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, $\{v_{t,s} \cdot \sum_{k=0}^{l-1} \alpha'_{\sigma_{t,k}} \cdot \alpha^{s+k}_{\sigma_{t,k}} \cdot z^k, v_{t,s}\}_{t=1,s=0}^{t=q,s=fs-1}$ to the set S. Moreover, the challenge query adds $\{v^*_{t,s} \cdot \sum_{k=0}^{l-1} \alpha'_{\sigma^*_{t,k}} \cdot \alpha^{s+k}_{\sigma^*_{t,k}} \cdot z^k, v^*_{t,s'}\}_{t=1,s=0,s\neq j+1,s'=0}^{t=q,s=fs-1,s'=fs-1}$.

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

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

So, according to Definition 2, the goal is to prove that one cannot find a combination of polynomials from R, S and T such that

$$(v_{j+1}^* \cdot \sum_{k=0}^{l-1} \alpha'_{\sigma^*_{\beta,k}} \cdot \alpha^{j+k+1}_{\sigma^*_{\beta,k}} \cdot z^k)(\sum_i u_i^a \cdot R^{(i)}) = \sum_{i,j} u_{i,j}^b \cdot R^{(i)} \cdot S^{(j)} + \sum_t u_t^{(c)} T^{(t)} \tag{4}$$

First let us note that the factor v_{j+1}^* only appears in the last element of the set S. 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 S will be involved in Equation 4. Moreover, since T is empty, the last sum of the Equation 4 can be omitted. Let $\{\vartheta_i^{(a)}, \vartheta_i^{(a)}\}_{i=0}^{i=fs-1}$ be constant scalars such that

$$(v^*_{j+1} \cdot \sum_{k=0}^{l-1} \alpha'_{\sigma^*_{\beta,k}} \cdot \alpha^{j+k+1}_{\sigma^*_{\beta,k}} \cdot z^k)(\sum_{i=0}^{fs-1} \vartheta^{(a)}_i \cdot z^i) = v^*_{j+1}(\sum_{i=0}^{fs-1} \vartheta^{(b)}_i \cdot z^i) \quad (5)$$

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 5, 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.

A.5 Proof of Theorem 5

As described in Definition 11, the correctness of the AS^{3}E construction depends mainly on the algorithms Encrypt, Issues, and Test. Since the instructions of the AS^{3}E's Encrypt and Issues algorithms are the same as in the S^{4}E construction (they differ only on the encryption key that is used) , and the fact that the AS^{3}E's Test algorithm is the same as we described for the S^{4}E construction. Then, we can use the proof of Theorem 1 to prove the AS^{3}E construction is correct.

A.6 Proof of Theorem 6

To prove the previous theorem, we use the same strategy as in the proof of Theorem 2. That is, we use a sequence of games $G^{(\beta)}_j$ for $j \in [1, n]$ where G^{β}_0 represents the game $Exp^{AS^3E\_T\_IND\_CPA}_{\mathcal{A},\beta}$ as defined in Definition 8. The idea is to show that the advantage of the adversary $\mathcal{A}$ for winning $Exp^{AS^3E\_T\_IND\_CPA}_{\mathcal{A},\beta}$ (i.e., G^{β}_0) is negligible.

By supposing 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 $Exp^{AS^3E\_T\_IND\_CPA}_{\mathcal{A},\beta}$ (Definition 8) and by following the same reasoning as in the proof of Theorem 2, we prove in the following Lemma that for all $j \in [0, n-1]$, for all $\beta \in \{0,1\}, |Pr[G^{\beta}_j(\lambda) = 1] - Pr[G^{\beta}_{j+1}(\lambda) = 1]|$ is negligible, which prove that $Adv^{Exp^{AS^3E\_T\_IND\_CPA}_{\mathcal{A},\beta}}(\lambda)$ is negligible.

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 s-GDH challenge containing $\{g^{z^i}, g^{\alpha'_\sigma \cdot (\alpha_\sigma \cdot z)^i}, g^{a_k \cdot z^i}, g^{\overline{a}_k \cdot z^i}\}_{i=0,\sigma \in \Sigma}^{i=2n-1}$ the simulator starts by generating the public key pk by defining $g^{z^i} = g^{z^{n-i^*+i}}$ (i.e., $g^{z^{i^*}} = g^{z^n}$), $g^{\alpha'_\sigma \cdot (\alpha_\sigma \cdot z)^i} = g^{\alpha'_\sigma \cdot (\alpha_\sigma \cdot z)^{n-i^*+i}}$. Then it defines $g_{i,F}$ 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 3a of $Exp_{\mathcal{A},\beta}^{E^4E\_T\_IND\_CPA}$ (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 $\overline{C'}_i = g^{\overline{a}_{\epsilon-1} z^{n+i_{\overline{F}_{\epsilon-1}} - i^*_{\overline{F}_{\epsilon-1}}}}$
  - $\forall i \in \mathcal{I}^{(j)} : C_i, \overline{C}_i \xleftarrow{\$} \mathbb{G}_1$
  - $\forall i \notin \mathcal{I}^{(j+1)}$ 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 $\overline{C'}_i = g^{\overline{a}_\epsilon z^{n+i_{\overline{F}_{\epsilon}} - i^*_{\overline{F}_{\epsilon}}}}$
  - $\forall i \in \mathcal{I}^{(j)} : C_i, \overline{C}_i \xleftarrow{\$} \mathbb{G}_1$
  - $\forall i \notin \mathcal{I}^{(j+1)}$ 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 $\overline{C'}_i = \emptyset$
  - $\forall i \in \mathcal{I}^{(j)} : C_i \xleftarrow{R} \mathbb{G}_1$ 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 $Exp_{\mathcal{A},\beta}^{E^4E\_T\_IND\_CPA}$ with non negligible advantage. By replacing $\alpha_{\sigma_{i^*}}$ by x_0 and $\alpha'_{\sigma_{i^*}}$ by x'_0, in order to 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 has already been proved the in Lemma 2.

A.7 Proof of Theorem 7

According to the definition 10, the adversary $\mathcal{A} = (\mathcal{A}_f, \mathcal{A}_g)$ is considered to have access to the the public parameters *params* and the public key $pk = \{g^{z^i}, g^{\alpha'_\sigma\cdot(\alpha_\sigma\cdot z)^i}\}_{i=0,\sigma\in\Sigma}^{i=fs-1}$. Following the a high min-entropy distribution, the part $\mathcal{A}_f$ of the adversary $\mathcal{A}$ chooses two patterns p_0^* and p_1^*. The challenger choose $\beta \in \{0,1\}$ randomly and creates $td_{p_\beta^*} = \{\widetilde{g}^{v_i}, \widetilde{g}^{V_i}\}_{i=0}^{i=fs-l-2}$ with $V_i = v_i \cdot \sum_{k=0}^{l-1}\alpha'_{\sigma_{p_\beta^*,k}}\cdot\alpha^{k+i}_{\sigma_{p_\beta^*,k}}\cdot z^k$ and k is number of symbols in p_β^*. The trapdoor $td_{p_\beta^*}$ is given to $\mathcal{A}_g$. We require $\mathcal{A}_f$ and $\mathcal{A}_g$ not to collude, since that would allow them to trivially break the rule indistinguishability property.

$\mathcal{A}_g$ is allowed to create any arbitrary chosen encrypted traffic using any pattern p. If the encrypted traffic contains p_i^*, then $\mathcal{A}_g$ can use the bilinear map e to easily determine whether the value of β chosen by the challenger equals to 0 or 1, however, this can happen only with negligible probability equal to $2^{-\mu(\lambda)}$, with $\mu(\lambda) \in \omega(\log(\lambda))$, since the pattern set has high-min entropy. Moreover, according to Theorem 6, AS^{3}E is trace indistinguishable meaning 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. Thus, the adversary $\mathcal{A}_g$ has to distinguish between $td_{p_\beta^*}$, $\beta \in \{0,1\}$ based on the content of the public key pk, the issued trapdoors tp_w, $w \in \mathcal{P}$ (Query phase of Definition 10). We show in the following that its hard under i-GDH assumption for $\mathcal{A}_g$ to distinguish $td_{p_0^*}$ and $td_{p_1^*}$.

We use the same strategy as in the proof of Theorem 4. Let $G_0^{(\beta)}$ denotes the $Exp_{\mathcal{A},\beta}^{P\_IND\_CPA}$ game, we will use a sequence of games $G_j^{(\beta)}, j \in [0, fs-1]$ to show that $\mathcal{A}_g$'s advantage is negligible. As in the proof of Theorem 4, 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 $G_j^{(\beta)}$ modifies $G_0^{(\beta)}$ 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_{fs-1}^{β}, the challenge trapdoor does not contain any useful information about w_β^*. As a result, the adversary cannot distinguish whether it plays $G_{fs-1}^{(0)}$ or $G_{fs-1}^{(1)}$. As a result, we can bound the advantage of $\mathcal{A}_g$ as following:

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

Then to prove that $Adv^{Exp^{P\_IND\_CPA}_{\mathcal{A}_g,\beta}}(\lambda)$ is negligible, we should prove that $\mathcal{A}$ cannot distinguish $G_j^{(\beta)}$ and $G_{j+1}^{(\beta)}$.

Let us consider fs to be the size of the fragment used by the AS^{3}E construction, and R, S, and T to be the three polynomial sets that are used in Definition 2. Let us suppose that the set of patterns $\mathcal{P}$ issued by $\mathcal{A}$ to $\mathcal{O}^S$ contains n_p patterns, and that the set $\mathcal{T}$ contains n_t traffics. According to the Issue algorithm of AS^{3}E, each issue query for the pattern $w_i = \sigma_{i,0} \cdots \sigma_{i,l-1}$ adds $\{v_{i,s} \cdot \sum_{k=0}^{l_i-1} \alpha'_{\sigma_{i,k}} \cdot \alpha_{\sigma_{i,k}}^{s+k} \cdot z^k, v_{i,s}\}_{i=0,s=0}^{i=q,s=fs-l_i-2}$ to the set S while the challenge query adds $\{v_s^* \cdot \sum_{k=0}^{l-1} \alpha'_{\sigma_k^*} \cdot \alpha_{\sigma_k^*}^{s+k} \cdot z^k, v_{s'}^*\}_{s=0,s\neq j+1,s'=0}^{s=fs-1,s'=fs-1}$. The set R will contain the elements of $\mathbb{G}_1$ that are provided in the public key pk. Thus $R = \{z^i, \alpha'_\sigma \cdot (\alpha_\sigma \cdot z)^i\}_{i=0,\sigma\in\Sigma}^{i=fs-1}$.

Therefore, to prove that $|Adv^{G_j^{(\beta)}}(\lambda) - Adv^{G_{j+1}^{(\beta)}}(\lambda)|$ is negligible under the i-GDH assumption, we should prove that one cannot find a combination of polynomials from R,S and T such that

$$\begin{aligned}(v_{j+1}^* \cdot \sum_{k=0}^{l-1} \alpha'_{\sigma^*_{\beta,k}} \cdot \alpha_{\sigma^*_{\beta,k}}^{j+k+1} \cdot z^k)(\sum_i \vartheta_i^a \cdot R^{(i)}) = \\ \sum_{i,j} \vartheta_{i,j}^b \cdot R^{(i)} \cdot S^{(j)} + \sum_t \vartheta_t^{(c)} T^{(t)}\end{aligned} \tag{6}$$

with constants ϑ_i^a and $\vartheta_{i,j}^a$ are constants.

First, we remark that the last sum in equation 6 can be remove since T is empty. Let

$$v^*_{j+1}\sum_{k=0}^{l-1}\alpha'_{\sigma^*_{\beta,k}}\cdot\alpha^{j+k+1}_{\sigma^*_{\beta,k}}\cdot z^k(\sum_{s=0}^{fs-1}(\vartheta^{(a)}_{j+1,s}\cdot z^s+\sum_{\sigma\in\Sigma}\vartheta^{(b)}_{j+1,s,\sigma}\ \alpha'_\sigma\cdot(\alpha_\sigma\cdot z)^s))=$$
$$(\sum_{t=0}^{q}\sum_{s=0}^{fs-1}(\vartheta^{(c)}_{t,s}\cdot v_{t,s}\sum_{k=0}^{l_i-1}\alpha'_{\sigma_{t,s+k}}\cdot\alpha^{s+k}_{\sigma_{t,s+k}}\cdot z^k)\ +\vartheta^{(d)}_{t,s}\cdot v_{t,s})\cdot$$
$$(\sum_{i=0}^{fs-1}\vartheta^{(e)}_{s,i}\cdot z^t+\sum_{\sigma\in\Sigma}\vartheta^{(f)}_{s,i,\sigma}\cdot\alpha'_\sigma\cdot(\alpha_\sigma\cdot z)^i)+$$
$$(\sum_{s=0,s\neq j+1}^{fs-1}(\vartheta^{(g)}_s\cdot v^*_s\sum_{k=0}^{l_i-1}\alpha'_{\sigma^*_{s+k}}\cdot\alpha^{s+k}_{\sigma^*_{s+k}}\cdot z^k)+\sum_{s=0}^{fs-1}\vartheta^{(h)}_s\cdot v^*_s)\cdot$$
$$(\sum_{i=0}^{fs-1}\vartheta^{(a)}_{s,i}\cdot z^i+\sum_{\sigma\in\Sigma}\vartheta^{(b)}_{s,i,\sigma}\cdot\alpha'_\sigma\cdot(\alpha_\sigma\cdot z)^i)$$

our goal is then to show that (i) $\forall s\in[0,fs-1]:\vartheta^{(a)}_{j+1,s}=0$ and (ii) $\forall s\in[0,fs-1],\forall\sigma\in\Sigma:\vartheta^{(b)}_{j+1,s,\sigma}=0$

Now, let us consider the previous each member of the previous equation as a polynomial in the variable v^*_{j+1}. we group the different monomials according to their degree in v^*_{j+1}.

$$v^*_{j+1}\sum_{k=0}^{l-1}\alpha'_{\sigma^*_{\beta,k}}\cdot\alpha^{j+k+1}_{\sigma^*_{\beta,k}}\cdot z^k(\sum_{s=0}^{fs-1}(\vartheta^{(a)}_{j+1,s}\cdot z^s+\sum_{\sigma\in\Sigma}\vartheta^{(b)}_{j+1,s,\sigma}\ \alpha'_\sigma\cdot(\alpha_\sigma\cdot z)^s))=$$
$$\vartheta^{(h)}_{j+1}\cdot v^*_{j+1}\cdot(\sum_{i=0}^{fs-1}\vartheta^{(a)}_{j+1,i}\cdot z^i+\sum_{\sigma\in\Sigma}\vartheta^{(b)}_{j+1,i,\sigma}\cdot\alpha'_\sigma\cdot(\alpha_\sigma\cdot z)^i)$$

We now consider each member in the previous equation as a polynomial in the variable α'. Then, we group the different monomials according to their degree in α'.

1. $\vartheta^{(h)}_{j+1}\cdot v^*_{j+1}\cdot\sum_{i=0}^{fs-1}\vartheta^{(a)}_{j+1,i}\cdot z^i=0$
2. $v^*_{j+1}\sum_{k=0}^{l-1}\alpha'_{\sigma^*_{\beta,k}}\cdot\alpha^{j+k+1}_{\sigma^*_{\beta,k}}\cdot z^k\cdot\sum_{s=0}^{fs-1}\vartheta^{(a)}_{j+1,s}\cdot z^s=\vartheta^{(h)}_{j+1}\cdot v^*_{j+1}\cdot\sum_{i=0}^{fs-1}\sum_{\sigma\in\Sigma}\vartheta^{(b)}_{j+1,i,\sigma}\cdot\alpha'_\sigma\cdot(\alpha_\sigma\cdot z)^i$
3. $v^*_{j+1}\sum_{k=0}^{l-1}\alpha'_{\sigma^*_{\beta,k}}\cdot\alpha^{j+k+1}_{\sigma^*_{\beta,k}}\cdot z^k\cdot\sum_{s=0}^{fs-1}\sum_{\sigma\in\Sigma}\vartheta^{(b)}_{j+1,s,\sigma}\ \alpha'_\sigma\cdot(\alpha_\sigma\cdot z)^s=0$

From Equation (3), since $\alpha'_\sigma,\alpha_\sigma,z,v^*$ are random scalars ($\forall\sigma\in\Sigma$), then $v^*_{j+1}\sum_{k=0}^{l-1}\alpha'_{\sigma^*_{\beta,k}}\cdot\alpha^{j+k+1}_{\sigma^*_{\beta,k}}\cdot z^k\neq0$ and $\sum_{s=0}^{fs-1}\sum_{\sigma\in\Sigma}\vartheta^{(b)}_{j+1,s,\sigma}\ \alpha'_\sigma\cdot(\alpha_\sigma\cdot z)^s=0$. From the latter we can

deduce that $\forall \sigma \in \Sigma, \forall s \in [0, fs-1], \vartheta^{(b)}_{j+1,s,\sigma} = 0$ which proved (ii).

Now, we regroup the element in the left side of Equation (2) and divide each element by v^*_{j+1} as following:

$$\sum_{k=0}^{fs+l-2} z^k \cdot \sum_{s=0}^{fs-1} \alpha'_{\sigma^*_{\beta,k-s}} \cdot \alpha^{j+k-s+1}_{\sigma^*_{\beta,k-s}} \cdot \vartheta^{(a)}_{j+1,s} = \vartheta^{(h)}_{j+1} \cdot \sum_{i=0}^{fs-1} \sum_{\sigma \in \Sigma} \vartheta^{(b)}_{j+1,i,\sigma} \cdot \alpha'_{\sigma} \cdot (\alpha_\sigma \cdot z)^i$$

Since the the length l of a pattern is greater or equal that 2, then $fs+l-2 \geq fs$. Let us now focus on the monomials of degree fs in z in the previous equation. In the left member, its coefficient is $\sum_{s=0}^{fs-1} \alpha'_{\sigma^*_{\beta,k-s}} \cdot \alpha^{j+k-s+1}_{\sigma^*_{\beta,k-s}} \cdot \vartheta^{(a)}_{j+1,s}$ and in the right member, its coefficient is 0 since the degree in z is at most $fs-1$. Therefore we have:

$$\sum_{s=0}^{fs-1} \alpha'_{\sigma^*_{\beta,k-s}} \cdot \alpha^{j+k-s+1}_{\sigma^*_{\beta,k-s}} \cdot \vartheta^{(a)}_{j+1,s} = 0$$

By considering the previous as polynomial in $\alpha_{\sigma^*_{\beta,k-s}}$, we have $\vartheta^{(a)}_{j+1,s} = 0$ for all $s \in [0, fs-1]$ which prove (i) and conclude the proof.