Efficient and secure generalized pattern matching via fast fourier transform

Damien Vergnaud

Efficient and Secure Generalized Pattern Matching *via* Fast Fourier Transform

Damien Vergnaud

École normale supérieure – C.N.R.S. – I.N.R.I.A.
45, Rue d'Ulm – 75230 Paris CEDEX 05 – France

Abstract. We present simple protocols for secure two-party computation of generalized pattern matching in the presence of malicious parties. The problem is to determine all positions in a text $\mathcal{T}$ where a pattern $\mathcal{P}$ occurs (or matches with few mismatches) allowing possibly both $\mathcal{T}$ and $\mathcal{P}$ to contain single character wildcards. We propose constant-round protocols that exhibit linear communication and quasilinear computational costs with simulation-based security. Our constructions rely on a well-known technique for pattern matching proposed by Fischer and Paterson in 1974 and based on the Fast Fourier Transform. The security of the new schemes is reduced to the semantic security of the ElGamal encryption scheme.

Keywords. Two-party secure computation, Pattern matching, Homomorphic encryption, Fast Fourier Transform

1 Introduction

We present secure protocols for two-party computation of (exact or approximate) pattern matching which are more efficient and conceptually simpler than previous approaches. Our proposals are constant-round and requires linear communication and quasilinear computational costs.

Prior work. The *pattern matching* problem [9] is to find all the occurrences of a given pattern $\mathcal{P}$ of length m in a text $\mathcal{T}$ of length n, both being sequences of characters drawn from a finite character set Σ. It is an important problem for many kinds of processes of strings, for instance in molecular biology, information retrieval, pattern recognition, compiling, data compression, program analysis and security. These applications often require more sophisticated forms of searching:

- *approximate pattern matching* [22], where the problem is to find the locations where the Hamming distance of $\mathcal{T}$ substrings and $\mathcal{P}$ is less than some threshold $k \leq m$;
- *pattern matching with wildcards* (or "*don't cares*") [11], where the problem is to find all occurrences of a pattern $\mathcal{P} \in (\Sigma \cup \{\star\})^m$ in a text $\mathcal{T} \in (\Sigma \cup \{\star\})^n$ where the wildcard character $\star \notin \Sigma$ matches any character in Σ.

An intensive research effort since the 1970s has led to the design of several efficient algorithms for generalized pattern matching (see [9]). In 1974, Fischer and Paterson [11] solved the pattern matching with wildcards problem in

$O(n \log m \log(\#\Sigma))$ time using convolution and *Fast Fourier Transform* (FFT). After several improvements, Clifford and Clifford [6] proposed a simple deterministic algorithm that also involves convolutions and runs in time $O(n \log m)$. A more complex algorithm in optimal $O(n)$ time was proposed in [20]. The FFT was also used to propose fast algorithms for the approximate pattern matching problem (*e.g.* [22]).

In the setting of secure *two-party computation*, introduced in 1982 by Yao [25], two parties wish to jointly compute some function of their private inputs while preserving a number of security properties. Troncoso-Pastoriza, Katzenbeisser and Celik [23] were the first to consider (basic) pattern matching in the context of secure computation (in the semi-honest setting). Their protocol implements the well-known Knuth-Morris-Pratt algorithm [19] and is linear in the input length. Hazay and Lindell [15] proposed later a protocol based on oblivious pseudorandom function evaluation which achieves one-sided simulatability security. The first construction for the (basic) pattern matching problem with full simulation-based security in the malicious setting was developed by Gennaro, Hazay and Sorensen in [12]. Their protocol relies on the Knuth-Morris-Pratt algorithm but requires linear round complexity and quadratic communication and computational cost. The first work which addresses the approximate pattern matching problem is [18].

Very recently, Hazay and Toft [17] proposed constant-rounds and efficient protocols for the (basic) pattern matching, the approximate pattern matching and the pattern matching with wildcards problems. Their schemes achieve security against malicious adversaries with (optimal) linear computational cost and bandwidth for the first problem but quadratic communication and computational complexity for the two extended problems.

Contributions of the paper. The main contribution of the paper is to provide protocols for approximate pattern matching (permitting a constant number of mismatches) and pattern matching with wildcards with constant-rounds, linear communication and $O(n \log m)$ computational costs.

	# of Rounds	Communication	Exponentiations
Basic Pattern Matching with alphabet Σ			
Hazay & Toft [17, §3]	$O(1)$	$O(n \log \lvert\Sigma\rvert)$	$O(n \log \lvert\Sigma\rvert)$
§3	$O(1)$	$O(n)$	$O(nm)$
Approximate Binary Pattern Matching with k Mismatches			
Hazay & Toft [17, §5]	$O(1)$	$O(nm)$	$O(nm)$
§5.1	$O(1)$	$O(nk)$	$O(n(\log m + k))$
Binary Pattern Matching with Wildcards			
Hazay & Toft [17, §4]	$O(1)$	$O(nm)$	$O(nm)$
§5.2	$O(1)$	$O(n)$	$O(n \log m)$

Table 1. Efficiency Comparison of Secure Generalized Pattern Matching Protocols

The problem we address can be described as follows: let us assume that Alice holds a text $\mathcal{T} \in (\Sigma \cup \{\star\})^n$, while Bob has a pattern $\mathcal{P} \in (\Sigma \cup \{\star\})^m$. The goal is for Bob to learn where $\mathcal{P}$ occurs (or matches with few mismatches) in $\mathcal{T}$ while Alice does not gain any information about $\mathcal{P}$ from the protocol execution and Bob does not learn anything but the matched text locations. Our approach relies on the classical *homomorphic encryption paradigm* (*e.g.* [16, § 7.2.2.]): Bob will encrypt $\mathcal{P}$ bit by bit for an additively homomorphic encryption scheme and Alice will then apply the deterministic algorithm from [6, 22] under encryption (using the homomorphic properties of the cryptosystem) and sends back the encrypted result to Bob.

First, we present a protocol for the (basic) pattern matching problem (§ 3) that is secure against malicious adversaries, but for the so-called *one-sided simulatability* weaker security notion. The construction relies on a straightforward observation that permits to perform secure pattern matching independently of the size of the alphabet Σ. Our main goal by presenting this scheme is to illustrate the ideas we will use in the following sections.

The Fast Fourier Transform (FFT) [7] is an algorithm to compute the discrete Fourier transform and its inverse. It leads notably to quasilinear polynomial multiplication algorithms. In order to apply Fischer and Paterson technique to secure two-party generalized pattern matching, one needs to construct a protocol for two-party FFT with quasilinear computational complexity. The idea of using FFT in secure two-party computation is probably not new[1] but we have not been able to locate a reference in the literature. We provide a description (§ 4) in the hope that it may be of independent interest.

Using this tool we provide adaptation of the generalized pattern matching algorithms from [6, 22, 2] and obtains schemes with overall efficiency summarized in the table **Tab. 1** (§ 5.1 and 5.2). As a by-product, we propose a protocol that reports the Hamming distance at every position (irrespective of its value) in $O(n\sqrt{m})$ time.

2 Preliminaries

2.1 Notation

For $n \in \mathbb{N}$, the set of n-bit strings is denoted by $\{0,1\}^n$, the set of integers $\{1, \ldots, n\}$ is denoted $[\![n]\!]$ and the symmetric group on $[\![n]\!]$ is denoted by $\mathfrak{S}_n$. The Hamming distance $d_H(x, y)$ between two words $x, y \in \{0,1\}^n$ is defined as the number of coordinates in which they differ.

We denote the security parameter by κ and input lengths are always assumed to be bounded by some polynomial in κ. A probabilistic algorithm is said to run in polynomial-time (PPT) if it runs in time that is polynomial in κ. Let $\mathcal{A}$ be a PPT algorithm and let x be an input for $\mathcal{A}$. The probability space that assigns to a string σ the probability that $\mathcal{A}$, on input x, outputs σ is denoted by $\mathcal{A}(x)$.

[1] An anonymous referee kindly informed us of the recent report [5] in which Cheon, Jarecki and Seo use FFT to speed-up secure set intersection protocols.

Given a probability space S, a PPT algorithm that samples a random element according to S is denoted by $x \xleftarrow{R} S$. For a finite set X, $x \xleftarrow{R} X$ denotes a PPT algorithm that samples a random element uniformly at random from X.

2.2 ElGamal Encryption

For a security parameter κ, the ElGamal encryption scheme [10] operates on a cyclic group $\mathbb{G}$ of κ-bits prime order q and identity element $1_{\mathbb{G}}$. Let g denote a generator of $\mathbb{G}$. The ElGamal public and secret keys are $(\mathbb{G}, q, g, y)$ and $(\mathbb{G}, q, g, x)$ (respectively) where x is picked uniformly at random in $\mathbb{Z}_q$ and $y = g^x$. A message $m \in \mathbb{G}$ is encrypted by picking r uniformly at random in $\mathbb{Z}_q$ and the ciphertext is $(g^r, y^r \cdot m)$. We will use the notation:

$$(g^r, y^r \cdot m) \xleftarrow{R} \mathsf{Enc}_y(m) \text{ or } (g^r, y^r \cdot m) \leftarrow \mathsf{Enc}_y(m; r)$$

A ciphertext (α, β) is decrypted as $m = \beta/\alpha^x \leftarrow \mathsf{Dec}_x(\alpha, \beta)$. The semantic security of the ElGamal encryption scheme follows from the hardness of Decision Diffie-Hellman (DDH) problem in $\mathbb{G}$ [24].

The ElGamal scheme is homomorphic relative to multiplication. In this paper, we consider a modified version of ElGamal where the encryption is performed in the exponents: one chooses r uniformly at random in $\mathbb{Z}_q$ and computes $(g^r, y^r \cdot g^m)$. Decryption of a ciphertext $c = (\alpha, \beta)$ is performed by computing $g^m = \beta/\alpha^x$. The fact that m cannot be efficiently recovered is not problematic for the way ElGamal is incorporated in our protocols. This variant of ElGamal is additively homomorphic and can be used to perform oblivious linear computations in the exponent: it naturally allows for multiplication with a plaintext constant using repeated doubling and adding.

2.3 Zero-Knowledge

Security in the presence of malicious behavior is usually achieved by forcing the parties to demonstrate that they are well-behaved. In our protocols, we need zero-knowledge proofs of knowledge that some algebraic statement $\mathcal{R}$ holds in a group $\mathbb{G} = \langle g \rangle$ of prime order q with $2^{\kappa-1} < q < 2^{\kappa}$. We will use Σ-protocols made secure against malicious verifiers (using standard techniques) since they are efficient and achieves constant communication complexity (see [17]):

$\boldsymbol{\pi}_{\mathbf{DL}}$: due to Schnorr [21], this Σ-protocol allows a prover to demonstrate knowledge of the solution to a discrete logarithm problem:

$$\mathcal{R}_{\mathrm{DL}} = \{[h, x] \mid h = g^x\}.$$

$\boldsymbol{\pi}_{\mathbf{eqDL}}$: due to Chaum and Pedersen [4], this Σ-protocol demonstrates equality of two discrete logarithm problems (as well as its knowledge):

$$\mathcal{R}_{\mathrm{eqDL}} = \{[(g_1, g_2, h_1, h_2), x] \mid h_1 = g_1^x \wedge h_2 = g_2^x\}.$$

We will use Σ-protocols for ElGamal ciphertexts for the public key $(\mathbb{G}, q, g, y)$:

$\boldsymbol{\pi_{\mathbf{Mult}}}$: due to Abe, Cramer and Fehr [1], this Σ-protocol demonstrates that a ciphertext is an encryption of the product of the plaintexts of two given ciphertexts:

$$\mathcal{R}_{\text{Mult}} = \{[(c_1, c_2, c_3), (m, r, s)] \,|c_1 = \mathsf{Enc}_y(g^m; r) \wedge c_3 = {c_2}^m \cdot \mathsf{Enc}_y(1_{\mathbb{G}}; s)\}.$$

$\boldsymbol{\pi_{\mathbf{Perm}}}$: due to Groth [13], this Σ-protocol demonstrates that a set of ciphertexts is a permutation and rerandomization of another set of ciphertexts:

$$\mathcal{R}_{\text{Perm}} = \left\{ [(c_i, \overline{c_i})_{i\in[\![k]\!]}, (\sigma, (r_i)_{i\in[\![k]\!]})] \,\middle|\, \begin{matrix} \sigma \in \mathfrak{S}_k \\ \overline{c_i} = c_{\sigma(i)} \cdot \mathsf{Enc}_y(1_{\mathbb{G}}; r_i), \forall i \in [\![k]\!] \end{matrix} \right\}.$$

$\boldsymbol{\pi_{\mathbf{nze}}}$: this Σ-protocol demonstrates that a ciphertext is a rerandomization of another ciphertext at some *non-zero* power:

$$\mathcal{R}_{\text{nze}} = \left\{ [(c_1, c_2), (R, r)] \,\middle|\, \quad c_1 = c_2^R \cdot \mathsf{Enc}_y(1_{\mathbb{G}}; r) \wedge R \neq 0 \right\}.$$

$\boldsymbol{\pi_{\mathbf{isBit}}}$: this Σ-protocol demonstrates that a ciphertext encrypts 0 or 1:

$$\mathcal{R}_{\text{isBit}} = \{[c, (b, r)] \mid c = \mathsf{Enc}_y(g^b; r) \wedge b \in \{0, 1\}\}.$$

$\boldsymbol{\pi_{\mathbf{isTrit}}}$: this Σ-protocol demonstrates that a ciphertext encrypts 0, 1 or 2:

$$\mathcal{R}_{\text{isTrit}} = \{[c, (b, r)] \mid c = \mathsf{Enc}_y(g^b; r) \wedge b \in \{0, 1, 2\}\}.$$

The protocol π_{nze} can be otained from π_{Mult} as described in [17] and the protocols π_{isBit} and π_{isTrit} can be obtained from π_{EqDL} using the technique of Cramer, Gennaro and Schoenmakers [8].

We also need a protocol π_{KeyGen} for generation of an ElGamal public key such that two parties hold shares of the secret key and a protocol π_{Dec} for shared decryption of a ciphertext encrypted using a key generated by π_{KeyGen}. We consider a protocol where only one party obtains the decrypted result (see [17, §2.3]). We denote the associated ideal functionalities $\mathcal{F}^{\text{KeyGen}}$ and $\mathcal{F}^{\text{Dec}}$.

2.4 Secure Two-Party Computation

We only provide a brief review of two-party computation definitions and we refer the reader to the recent book [16, Chapter 2] for more details. Let f be a two-argument function and let Alice and Bob be two possibly malicious parties, the first having an input x and the second having an input y. Securely computing f means that Alice and Bob keep turns exchanging message strings so that:

- Bob learns $z = f(x, y)$ but nothing about x (not already implied by y and z);
- Alice learns nothing about y (and nothing about z not already implied by x).

In particular, Alice and Bob wish to ensure that nothing is revealed from the protocol execution to an outsider or the other party (*privacy*) and that the output is computed according to the specified function (*correctness*). In this paper, we do not require the correctness to hold with probability 1 but instead allow negligible probability of error in the value output by the protocol.

The *simulation-based* security definitions formalize the intuition that whatever can be computed by a party can be computed based on its input and output only. This is done by comparing a party's view in a real protocol execution to an "ideal execution", where a trusted party computes the function and sends the output to the parties. In this paper, we consider two flavors of simulation-based security in the presence of malicious adversaries:

- *Full simulation security* [3]: it requires that for every PPT adversary $\mathcal{A}$ in the real world, there exists a corresponding PPT simulator $\mathcal{S}$ in the ideal world such that the view are computationally indistinguishable.
- *One-sided simulation security*: where full simulation is provided for only one of the corruption cases and only privacy is achieved for the other case (guaranteeing that the adversary does not learn anything but the output of the computation).

3 Warm Up: (Basic) Pattern Matching Protocol

In this section, we address the question of how to securely compute the functionality $\mathcal{F}_{\mathrm{PM}}$ defined by

$$((\mathcal{P}, n), (\mathcal{T}, m)) \mapsto \begin{cases} (\{j \in [\![n-m+1]\!], \mathcal{T}_j = \mathcal{P}\}, \perp) & \text{if } \mathcal{P} \in \Sigma^m \wedge \mathcal{T} \in \Sigma^n \\ (\perp, \perp) & \text{otherwise} \end{cases}$$

where $\mathcal{T}_j$ denotes the substring of length m that begins at the j-th position in $\mathcal{T}$. Note that Alice learns nothing about $\mathcal{P}$ and the only thing Bob learns about $\mathcal{T}$ is the locations where $\mathcal{P}$ appears. We assume that $\#\Sigma = \mathsf{poly}(\kappa)$ and $\Sigma \subseteq \mathbb{Z}_q$.

Motivated by the task of constructing a simple protocol for $\mathcal{F}_{\mathrm{PM}}$, we use the straightforward observation that $\mathcal{P} = (p_1, \ldots, p_m)$ occurs at the j-th position in $\mathcal{T} = (t_1, \ldots, t_n)$ if and only if $p_i - t_{i+j-1} = 0$ for all $i \in [\![m]\!]$ and that consequently for random $(s_{1,j}, \ldots, s_{m,j}) \in \mathbb{Z}_q^m$, the equality

$$\sum_{i=1}^{m} s_{i,j}(p_i - t_{i+j-1}) = 0 \tag{1}$$

holds with probability 1 if $\mathcal{P} = \mathcal{T}_j$ and with probability $1/q$ otherwise (where the probability is taken over the random $s_{i,j}$ for $i \in [\![m]\!]$).

This observation results in a simple protocol for $\mathcal{F}_{\mathrm{PM}}$ that can be summarized as follows:

1. Bob picks at random $x \in \mathbb{Z}_q$ and sets $y = g^x$. He sends y to Alice and proves his knowledge of x by running π_{DL} using x.

2. Bob encrypts $\mathcal{P}$ character by character with his own public key y:

$$\boldsymbol{c_i} = (\alpha_i, \beta_i) \xleftarrow{R} \mathsf{Enc}_y(g^{p_i}) \quad \text{for } i \in [\![m]\!]$$

and sends $\boldsymbol{c_i}$ to Alice for $i \in [\![m]\!]$.

3. Using the homomorphic property of ElGamal encryption, Alice computes, for $j \in [\![n-m+1]\!]$, a (uniformly distributed) encryption of the left-hand side of (1) as

$$\begin{aligned}\boldsymbol{d_j} &= \left(g^{r_j}\prod_{i=1}^{m}\alpha_i^{s_{i,j}}, y^{r_j}\prod_{i=1}^{m}\left(\beta_i g^{-t_{j+i-1}}\right)^{s_{i,j}}\right)\\ &= \mathsf{Enc}_y(1_{\mathbb{G}}, r_j)\cdot\prod_{i=1}^{m}\left(\boldsymbol{c_i}\cdot(1_{\mathbb{G}}, g^{-t_{j+i-1}})\right)^{s_{i,j}}\end{aligned}$$

for $r_j, s_{1,j}, \ldots, s_{m,j}$ picked uniformly at random in $\mathbb{Z}_q$. Alice sends $\boldsymbol{d_j}$ for $j \in [\![n-m+1]\!]$ to Bob.

4. Bob decrypts $\boldsymbol{d_j}$ for $j \in [\![n-m+1]\!]$ thanks to his knowledge of x and outputs the set of indices of ciphertexts that encrypt $1_{\mathbb{G}}$.

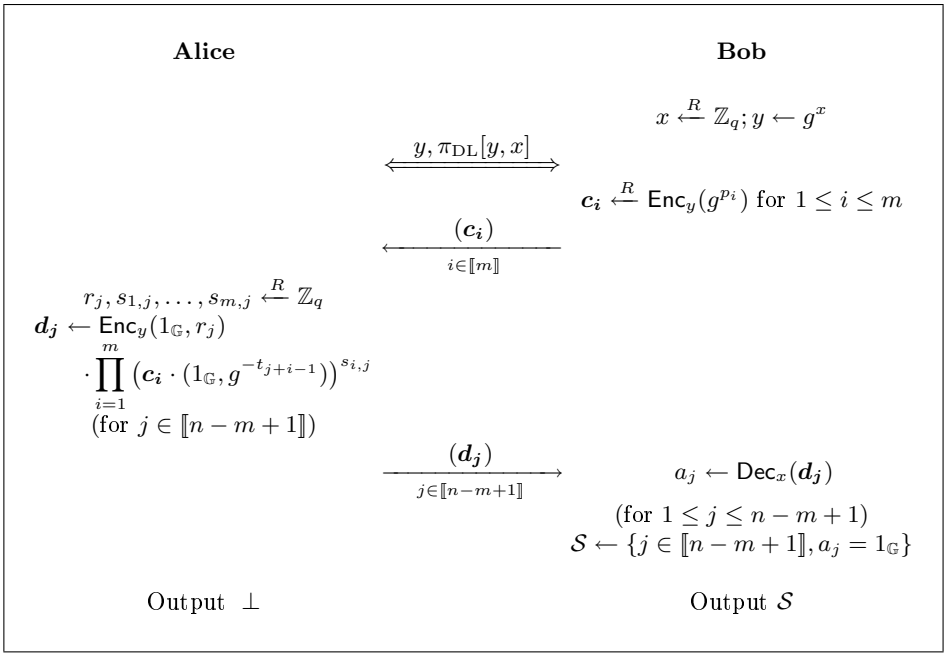

Fig. 1: Secure Basic Pattern Matching

The protocol flow is described in **Fig. 1** (where simple arrows indicate data transmission and double arrows stand for interactive protocols). The protocol is

less efficient than the scheme from [17] since it exhibits linear communication[2] but quadratic computation costs. However it is simpler and illustrates the ideas we will use in the next sections. It achieves one-sided simulation security against malicious adversaries.

Theorem 1. *If π_{DL} is a zero-knowledge proof of knowledge of a discrete logarithm secure against malicious verifiers and if the ElGamal encryption scheme is semantically secure, then the protocol described in* ***Fig. 1*** *securely computes $\mathcal{F}_{\mathrm{PM}}$ with one-sided simulation security against malicious adversaries.*

Proof (Sketch). As in [17, Theorem 1], we separately prove the security in the case that Alice is corrupted and the case that Bob is corrupted.

Alice is corrupted: Since Alice does not receive any output from the execution, and we are only proving one-sided simulation security here, all we need to show is that privacy is preserved. This follows readily from the semantic security of the ElGamal encryption scheme.

Bob is corrupted: Let $\mathcal{A}$ denote an adversary controlling Bob. We need to prove that $\mathcal{A}$ does not learn anything but the matching text locations. We construct a simulator $\mathcal{S}$ as follows:

1. $\mathcal{S}$ is given a pattern of length m, an integer n and $\mathcal{A}$'s auxiliary input and invokes $\mathcal{A}$ on these values.
2. $\mathcal{S}$ obtains $\mathcal{A}$'s secret key x from the proof of knowledge extractor for π_{DL}.
3. $\mathcal{S}$ receives from the adversary $\mathcal{A}$ a vector of m ciphertexts $\boldsymbol{c_i}$ for $i \in [\![m]\!]$. Using x, it decrypts each ciphertext and try to compute the discrete log of the corresponding plaintext z_i.
   (a) If $z_i = g^{\sigma_i}$ where $\sigma_i \in \Sigma$ for all $i \in [\![m]\!]$, $\mathcal{S}$ defines the pattern $\mathcal{P}$ as $\mathcal{P} = (\sigma_1, \ldots, \sigma_m)$ and sends it to the trusted party for $\mathcal{F}_{PM}$ and obtains a subset $\mathcal{I} \subseteq [\![n-m+1]\!]$. The simulator $\mathcal{S}$ produces a vector of ciphertexts:
   $$\boldsymbol{d_j} \xleftarrow{R} \left(\mathsf{Enc}_y(g^{b_j})\right)^{\theta_j} \text{ for } j \in [\![n-m+1]\!]$$
   where $b_j \in \{0,1\}$ with $b_j = 0 \iff j \in \mathcal{I}$ and $\theta_j \xleftarrow{R} \mathbb{Z}_q$.
   (b) If there is an $i \in [\![m]\!]$ such $z_i \neq g^{\sigma_i}$ for all $\sigma_i \in \Sigma$, then $\mathcal{S}$ aborts by sending $\perp$ to the trusted party for $\mathcal{F}_{PM}$ and produces a vector of ciphertexts:
   $$\boldsymbol{d_j} \xleftarrow{R} \left(\mathsf{Enc}_y(g)\right)^{\theta_j} \text{ for } j \in [\![n-m+1]\!]$$
   where $\theta_j \xleftarrow{R} \mathbb{Z}_q$.
4. If at any point $\mathcal{A}$ sends an invalid message $\mathcal{S}$ aborts, sending $\perp$ to the trusted party for $\mathcal{F}_{PM}$. Otherwise, it outputs whatever $\mathcal{A}$ does.

[2] The round complexity and the communication complexity of our scheme are however smaller by a constant factor and independent of the size of the underlying alphabet.

Since $|\Sigma| = \mathsf{poly}(\kappa)$, the simulator $\mathcal{S}$ runs in polynomial time (the running time of the third step being upper-bounded by $O(m\sqrt{|\Sigma|})$ group operations using generic technique for discrete logarithm computation in short intervals). Our basic observation on (1) shows readily that the adversary's view is statistically close to its view in the real execution of the protocol. □

Remark 1. Observe that the protocol from **Fig. 1** does not achieve correctness when Alice is corrupted. However, it is possible to achieve full simulation security against malicious adversaries by enforcing Alice to use only text symbols t_i in Σ for $i \in [\![n]\!]$ (using for instance generalization of π_{isBit} and π_{isTrit}) and proving consistency of the ciphertexts $(\delta_j, \gamma_j)_{1 \leq j \leq n-m+1}$.

4 Secure Fast Fourier Transform and Polynomial Multiplication

4.1 FFT and Polynomial Multiplication: a Brief Recall

Let q be a prime number, D a divisor of $q-1$ and ω a primitive D-th root of 1 in $\mathbb{Z}_q^*$ (for simplicity, one can assume that D is a power of two). Suppose we are given two polynomials in $A, B \in \mathbb{Z}_q[X]$ of degree less than $d = D/2$. The FFT is a well-known method to compute the coefficients of $C(X) = A(X)B(X)$ in $O(D \log D)$ multiplications in $\mathbb{Z}_q$:

1. Evaluate A and B at the D points: 1, ω, ω^2, ..., ω^{D-1} using the Discrete Fourier Transform (DFT):

> **Evaluation of** $P \in \mathbb{Z}_q[X]$ **at** 1, ω, ω^2, **...**, ω^{D-1}
> Write $P(X) = P_0(X^2) + XP_1(X^2)$
> Evaluate (recursively) P_0 and P_1 at $1, \omega^2, \omega^4, \ldots, \omega^{D-1}$
> Write $P(\omega^i) = P_0(\omega^{2i}) + \omega^i P_1(\omega^{2i})$ for $i \in [\![D]\!]$

2. Compute the values of $C(X)$ at these D points 1, ω, ω^2, ..., ω^{D-1}.

3. Interpolate the polynomial C using the inverse DFT:

> **Interpolation of** $P \in \mathbb{Z}_q[X]$ **given values at** 1, ω, ω^2, **...**, ω^{D-1}
> Write $\widetilde{P}(X) = P(1) + P(\omega)X + \cdots + P(\omega^{D-1})X^{D-1}$
> Evaluate (as in **1.**) $\widetilde{P}$ at 1, ω^{-1}, ω^{-2}, ..., ω^{1-D}
> Output $P(X) = \dfrac{1}{D}\left(\widetilde{P}(1) + \widetilde{P}(\omega^{-1})X + \cdots + \widetilde{P}(\omega^{1-D})X^{D-1}\right)$

4.2 Secure Fast Polynomial Multiplication

We now outline a protocol to efficiently and securely compute the encryption of the product of two polynomials (for a given public key y). Let us assume that Alice holds a polynomial $A(X) = \sum_{i=0}^{d-1} a_i X^i \in \mathbb{Z}_q[X]$ and that Bob holds a polynomial $B(X) = \sum_{i=0}^{d-1} b_i X^i \in \mathbb{Z}_q[X]$ both of degree less than d.

1. Alice encrypts the polynomial A coefficient by coefficient (in the exponents):
$$\alpha_0 = \mathsf{Enc}_y(g^{a_0}), \alpha_1 = \mathsf{Enc}_y(g^{a_1}), \ldots, \alpha_{d-1} = \mathsf{Enc}_y(g^{a_{d-1}})$$
and sends the ciphertexts $(\alpha_0, \alpha_1, \ldots, \alpha_{d-1})$ to Bob.
2. Bob encrypts the polynomial B coefficient by coefficient (in the exponents):
$$\beta_0 = \mathsf{Enc}_y(g^{b_0}), \beta_1 = \mathsf{Enc}_y(g^{b_1}), \ldots, \beta_{d-1} = \mathsf{Enc}_y(g^{b_{d-1}})$$
and sends the ciphertexts $(\beta_0, \beta_1, \ldots, \beta_{d-1})$ to Alice.
3. Alice and Bob (simultaneously and independently) compute the encrypted value of $g^{A(\omega^i)}$ and $g^{B(\omega^i)}$ for $i \in [\![D]\!]$ using the DFT (**Step 1** in the FFT method). Since the encryption scheme is additively homomorphic, it can be used to perform deterministic oblivious linear computations in the exponent. At this step of the protocol, Alice and Bob then share (identical) encryption of $g^{A(\omega^i)}$ and $g^{B(\omega^i)}$ for $i \in [\![D]\!]$ and Alice[3] knows the randomness corresponding to the encryption of $g^{A(\omega^i)}$.
4. Alice computes the encryption of $g^{A(\omega^i)B(\omega^i)}$ for $i \in [\![D]\!]$ knowing $A(\omega^i)$ and the shared encryption of $g^{B(\omega^i)}$ (**Step 2** in the FFT method). Alice sends the resulting ciphertexts to Bob and, thanks to her knowledge of the randomness in the shared encryption of $g^{A(\omega^i)}$ she can prove its validity by running the protocol π_{Mult}.
5. Alice and Bob (simultaneously and independently) compute the encrypted coefficients of AB (in the exponents) using the inverse DFT (**Step 3** in the FFT method).

The computational complexity of the protocol is $O(D \log D)$ exponentiations in $\mathbb{G}$ and its communication complexity is $O(D)$ (and therefore optimal). In order to use this fast two-party polynomial multiplication protocol in our generalized pattern matching schemes, we will need the encrypted computation to be done under a public-key whose corresponding secret-key is shared between Alice and Bob. The fact that intermediate values are encrypted under a key which neither Alice or Bob know permits the zero-knowledge simulation.

We do not propose any specific application in the realm of polynomial arithmetic for this protocol but we will rather provide applications to the generalized pattern matching problem in the following section.

Remark 2. It is possible to enforce the polynomials coefficients to belong to a specific subset of $\mathbb{Z}_q$ (*e.g.* we will use the set $\{0,1\}$ or $\{0,1,2\}$ in the following and use the protocols π_{isBit} and π_{isTrit} to ensure these properties).

5 Fast Secure Generalized Pattern Matching

The main idea of the algorithm from [6] is to calculate the sum of squared differences between the pattern and the text for every possible alignment. Suppose

[3] Of course, Bob knows the randomness corresponding to the encryption of $g^{B(\omega^i)}$ and in the rest of the protocol, one can exchange the role of Alice and Bob.

without loss of generality that $\Sigma \subset \mathbb{N}$. If there are no wildcards then for each location $i \in [\![n-m+1]\!]$, we can calculate

$$\sum_{j=1}^{m}(p_j - t_{i+j-1})^2 = \sum_{j=1}^{m}\left({p_j}^2 - 2p_j \cdot t_{i+j-1} + {t_{i+j-1}}^2\right) \tag{2}$$

in $O(n \log n)$ time using FFT. Wherever there is an exact match this sum will be exactly 0. When wildcards are allowed in the pattern and the text we replace the wildcard symbols by 0's (and other symbols by non-negative integers) and then consider the sum

$$\sum_{j=1}^{m} p_j t_{i+j-1}(p_j - t_{i+j-1})^2 = \sum_{j=1}^{m}\left({p_j}^3 t_{i+j-1} - 2{p_j}^2 \cdot {t_{i+j-1}}^2 + p_j {t_{i+j-1}}^3\right) \tag{3}$$

which equals 0 if and only if there is an exact match with wildcards. The sum (3) can be computed with three convolutions and therefore in $O(n \log n)$ time using FFT.

In order to reduce the time complexity from $O(n \log n)$ to $O(n \log m)$, we will use a standard trick which consists in partitioning $\mathcal{T}$ into n/m overlapping substrings of length $2m$ with the first substring starting at the beginning of the text and each subsequent substring having an overlap of length m with the previous one. The matching algorithm is then performed separately on each substring. Each iteration takes $O(m \log m)$ time giving an overall time complexity of $O((n/m)m \log m) = O(n \log m)$.

In the following, we will use this trick and for simplicity we will assume (without loss of generality) that $n = 2m$ and that $4m$ is a power of 2 dividing $q - 1$. The protocol for (fast) secure polynomial multiplication can be used to securely compute sums of the form (2) or (3) in time $O(m \log m)$. The figure **Fig. 2** presents the common opening for our generalized pattern matching protocols. At the end of this subprotocol, Alice and Bob hold

- shares (x_a, x_b) of the secret key corresponding to the public key y.
- encryption $\boldsymbol{a_i}$ of the **bit** t_i for $i \in [\![2m]\!]$.
- encryption $\boldsymbol{b_i}$ of the **bit** p_{m-i+1} for $i \in [\![m]\!]$.
- encryption $\boldsymbol{f_{i+m}}$ of the sum $\sum_{j=1}^{m} p_j \cdot t_{i+j-1}$ for $i \in [\![m]\!]$.

Note that we modify the ordering of the pattern bits, in order to compute (2) or (3) as an actual convolution and that we enforce ciphertexts $\boldsymbol{a_i}$ and $\boldsymbol{b_i}$ to encrypt bits using the protocol π_{isBit} (as mentioned above).

5.1 Pattern Matching with Mismatches

Our first algorithm is in fact independent of the bound k and report the Hamming distance at every position irrespective of its value.

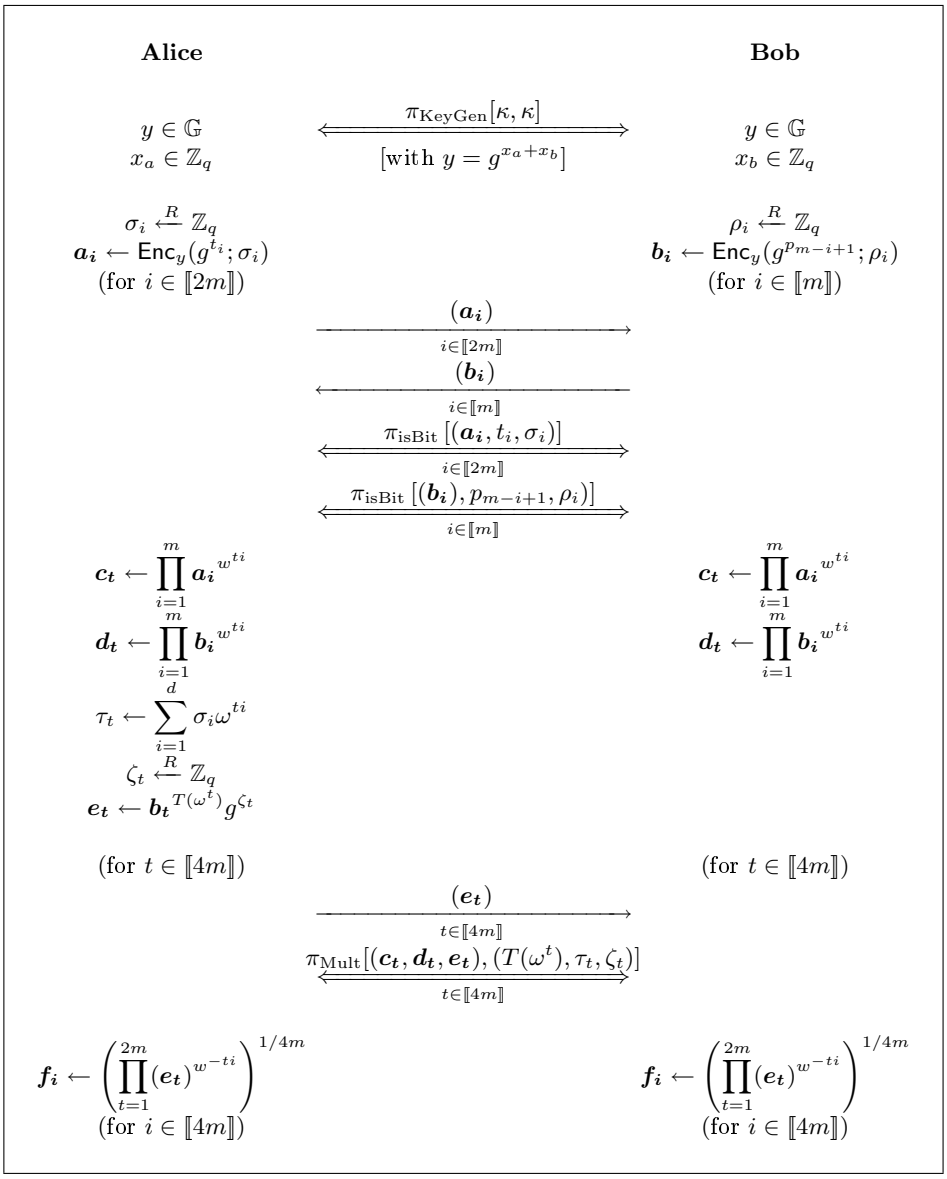

Fig. 2: Basic Protocol for Secure Generalized Pattern Matching

Secure Hamming Distance Computation. The paper [18] examined secure two-party computation of functions which depend only on the Hamming distance of the inputs of the two parties. We revisit this problem in this paragraph and propose an efficient protocol for the following problem: given a binary text $\mathcal{T}$ of length n and a binary pattern $\mathcal{P}$ of length m, compute the Hamming distance between $\mathcal{P}$ and $\mathcal{T}_i$ for $i \in [\![n - m + 1]\!]$, *i.e.* we consider the question of how to

compute the functionality, denoted by $\mathcal{F}_{\text{HD}}$:

$$((\mathcal{P}, n), (\mathcal{T}, m)) \mapsto \left(\left(d_H(\mathcal{T}_i, \mathcal{P}) \right)_{i \in [\![n-m+1]\!]}, \perp \right)$$

Again Alice learns nothing about $\mathcal{P}$ and the only thing that Bob learns about $\mathcal{T}$ is the number of matches of $\mathcal{P}$ and $\mathcal{T}_i$ for $i \in [\![n-m+1]\!]$. We denote $\text{HD}(\mathcal{P}, \mathcal{T})$ the *Hamming distance vector of* $\mathcal{P}$ *and* $\mathcal{T}$ output by this functionality.

Over a binary alphabet, the sum (2) becomes:

$$d_H(\mathcal{T}_i, \mathcal{P}) = \sum_{j=1}^{m} \left({p_j}^2 - 2p_j \cdot t_{i+j} + {t_{i+j}}^2 \right) = \sum_{j=1}^{m} \left(p_j - 2p_j \cdot t_{i+j} + t_{i+j} \right) \quad (4)$$

and therefore, with the previous notation, the product $\prod_{j=1}^{m} \boldsymbol{a_{i+j}} \cdot \boldsymbol{b_j} \cdot {\boldsymbol{f_{i+m}}}^{-2}$ encrypts the value $d_H(\mathcal{T}_i, \mathcal{P})$ for $i \in [\![n-m+1]\!]$. It is therefore easy to extend the opening of **Fig. 2** in order to securely compute the functionality $\mathcal{F}_{\text{HD}}$. The detailed protocol flow is given in **Fig. 3**.

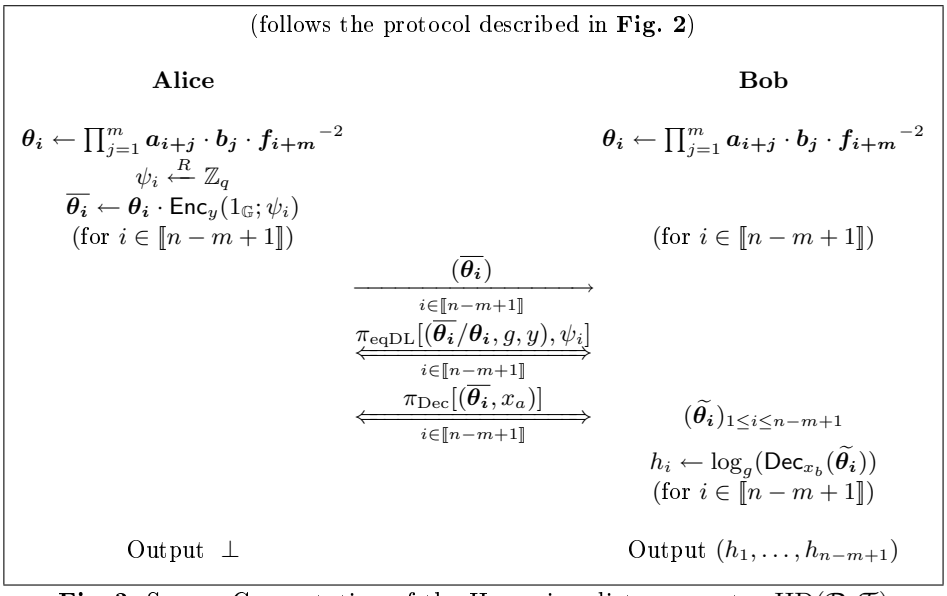

Fig. 3: Secure Computation of the Hamming distance vector $\text{HD}(\mathcal{P}, \mathcal{T})$

The protocol is constant-round with linear communication complexity and $O(n\sqrt{m})$ time complexity. We can prove that it achieves full simulation security against malicious adversaries.

Theorem 2. *If* π_{eqDL}, π_{Mult}, π_{isBit} *are zero-knowledge proofs of knowledge secure against malicious verifiers for the languages* $\mathcal{R}_{\text{eqDL}}$, $\mathcal{R}_{\text{Mult}}$ *and* $\mathcal{R}_{\text{isBit}}$, *if*

π_{KeyGen} and π_{Dec} are protocols secure against malicious verifiers for the functionality $\mathcal{F}^{\text{KeyGen}}$ and $\mathcal{F}^{\text{Dec}}$ and if the ElGamal encryption scheme is semantically secure, then the protocol described in ***Fig. 3*** *securely computes $\mathcal{F}_{\text{HD}}$ in the presence of malicious adversaries.*

Proof (Sketch). We consider only the security in the case that Bob is corrupted (the proof in the case that Alice is corrupted follows the same lines).

Bob is corrupted: Let $\mathcal{A}$ denote an adversary controlling Bob. We need to prove that $\mathcal{A}$ does not learn anything but the matching text locations. We construct a simulator $\mathcal{S}$ as follows:

1. $\mathcal{S}$ is given a pattern of length m, an integer n and $\mathcal{A}$'s auxiliary input and invokes $\mathcal{A}$ on these values.
2. $\mathcal{S}$ emulates the trusted party for π_{KeyGen} as follows by choosing two random elements x_A and x_B in $\mathbb{Z}_q$ and hands $\mathcal{A}$, its share x_B and the public key $y = g^{x_A + x_B}$.
3. $\mathcal{S}$ receives from the adversary $\mathcal{A}$ a vector of m ciphertexts (α_i, β_i) for $i \in [\![m]\!]$. If the functionality $\mathcal{F}_{ZK}^{\mathcal{R}_{isBit}}$ aborts, then $\mathcal{S}$ aborts, sending $\perp$ to the trusted party for $\mathcal{F}_{HW}$.
4. Otherwise $\mathcal{S}$ defines the pattern $\mathcal{P}$ using the witnesses for π_{isBit} and sends $\mathcal{P}$ to the ideal functionality $\mathcal{F}_{HW}$.
5. The simulator $\mathcal{S}$ sets $\mathcal{T} = 1^n \in \Sigma^n$.
6. $\mathcal{S}$ completes the execution as the honest Alice would on input $\mathcal{T}$ but simulates the proof in order to be consistent with the vector output by the ideal functionality $\mathcal{F}_{HW}$. If at any point $\mathcal{A}$ sends an invalid message $\mathcal{S}$ aborts, sending $\perp$ to the trusted party for $\mathcal{F}_{PM}$. Otherwise, it outputs whatever $\mathcal{A}$ does.

Following the proof of [17, Theorem 1], it is easy to prove that if there exists a distinguisher $\mathcal{D}$ for these executions, then there exist a PPT adversary breaking the semantic security of ElGamal encryption. □

Secure Pattern Matching with k Mismatches. We now propose an efficient protocol for the following problem: given an integer k, a binary text $\mathcal{T}$ of length n and a binary pattern $\mathcal{P}$ of length m, compute the functionality, denoted by $\mathcal{F}_{\text{PM}-k}$:

$$((\mathcal{P}, n), (\mathcal{T}, m)) \mapsto (\{i \in [\![n - m + 1]\!], d_H(\mathcal{T}_i, \mathcal{P}) \leq k\}, \perp)$$

Again Alice learns nothing about $\mathcal{P}$ and the only thing that Bob learns about $\mathcal{T}$ is the locations where $\mathcal{P}$ appears with less than k mismatches.

Let us recall that the protocol of **Fig. 3** produces ciphertexts $\boldsymbol{\theta_i}$ that encrypt the value $d_H(\mathcal{T}_i, \mathcal{P})$ for $i \in [\![n - m + 1]\!]$. Therefore

$$[\boldsymbol{\theta_i} \cdot (1_{\mathbb{G}}, g^{-\ell})]^u \cdot \mathsf{Enc}_y(1_{\mathbb{G}}, v) \tag{5}$$

for random $u \xleftarrow{R} \mathbb{Z}_q^*$ and $v \xleftarrow{R} \mathbb{Z}_q$ is a uniformly distributed encryption of $1_{\mathbb{G}}$ if $d_H(\mathcal{T}_i, \mathcal{P}) = \ell$ and of a random element from $\mathbb{G}$ otherwise.

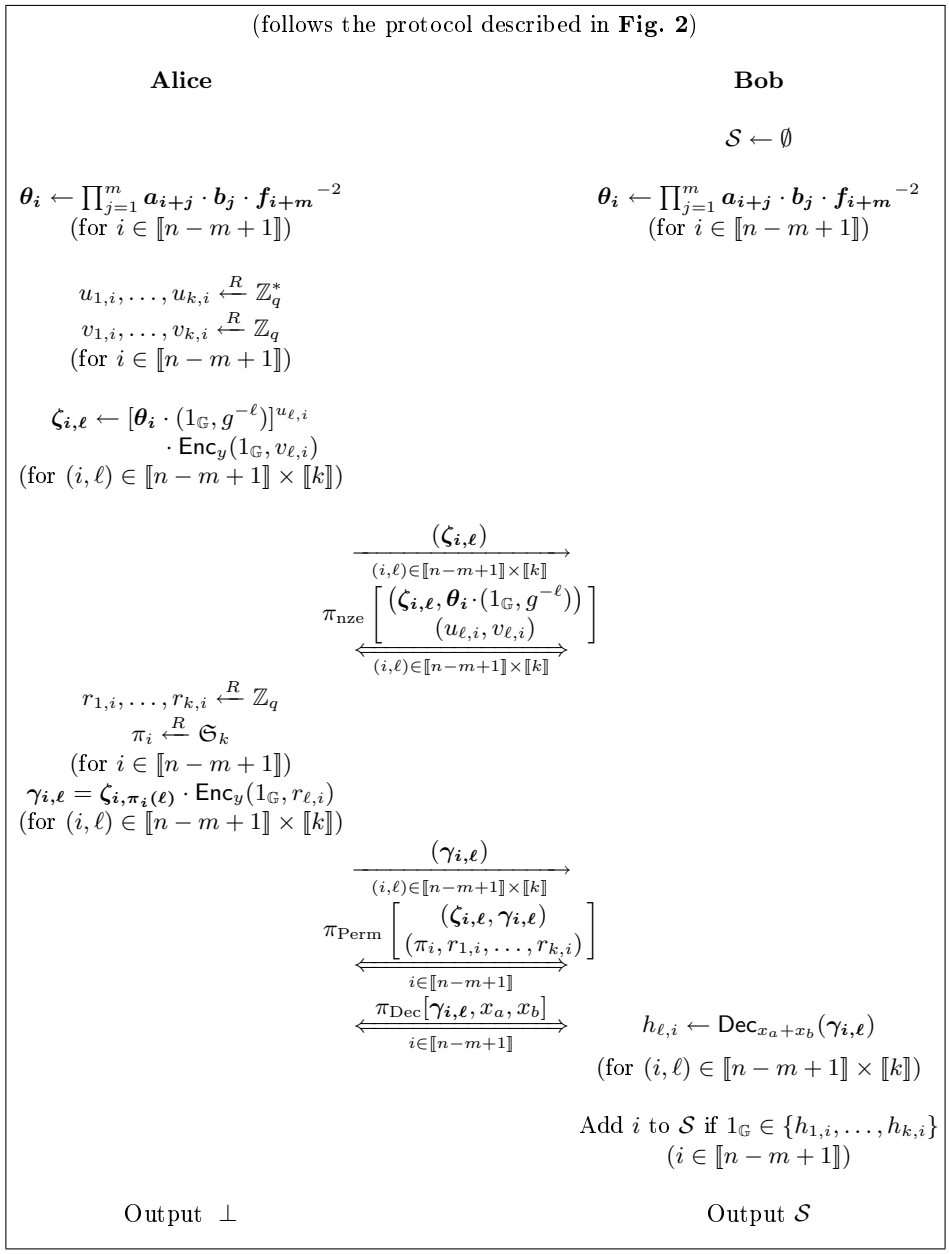

Fig. 4: End of the Protocol for Secure Pattern Matching with Mismatches

It is therefore easy to extend the opening of **Fig. 2** in order to securely compute the functionality $\mathcal{F}_{\mathrm{PM}-k}$: Alice pick, for all $i \in [\![n-m+1]\!]$, a permutation $\sigma \in \mathfrak{S}_k$ and permute and rerandomize the ciphertexts from (5). She proves for all $i \in [\![n-m+1]\!]$ the correctness of this operation by running π_{Perm} on k ElGa-

mal ciphertexts. As in the previous protocol, Alice then partially decrypts the permuted ciphertexts for all $i \in [\![n-m+1]\!]$ by running all π_{Dec} on private input x_A. Bob then finishes the decryption and outputs the set of $i \in [\![n-m+1]\!]$ such that the vector of (permuted) at location i contains one (and only one) encryption of $1_{\mathbb{G}}$. The detailed protocol flow is given in **Fig. 4**.
We have the following theorem whose proof will be given in the full version of the paper:

Theorem 3. *If π_{eqDL}, π_{Mult}, π_{Perm}, π_{nze} and π_{isBit} are zero-knowledge proofs of knowledge secure against malicious verifiers for the languages $\mathcal{R}_{\text{eqDL}}$, $\mathcal{R}_{\text{Mult}}$, $\mathcal{R}_{\text{Perm}}$, $\mathcal{R}_{\text{nze}}$ and $\mathcal{R}_{\text{isBit}}$, if π_{KeyGen} and π_{Dec} are protocols secure against malicious verifiers for the functionality $\mathcal{F}^{\text{KeyGen}}$ and $\mathcal{F}^{\text{Dec}}$ and if the ElGamal encryption scheme is semantically secure, then the protocol described in **Fig. 4** securely computes $\mathcal{F}_{\text{PM}-k}$ in the presence of malicious adversaries.*

5.2 Pattern Matching with Wildcards

Finally, we consider the secure computation of the functionality $\mathcal{F}_{\text{PM}\star}$ that on input $((\mathcal{P}, n), (\mathcal{T}, m))$ computes

$$\begin{cases} (\{j \in [\![n-m+1]\!] \mid \forall i \in [\![m]\!], p_i = \star \vee t_{i+j-1} = \star \vee p_i = t_{i+j-1}\}, \bot) \\ & \text{if } \mathcal{P} \in \{\star, 0, 1\}^m \wedge \mathcal{T} \in \{\star, 0, 1\}^n \\ (\bot, \bot) & \text{otherwise} \end{cases}$$

Again, Alice learns nothing about $\mathcal{P}$ and the only thing Bob learns about $\mathcal{T}$ is the locations where $\mathcal{P}$ appears (with wildcards).

The protocol is similar to the previous ones and relies on the fact that the sum (3) can be computed with three convolutions and therefore in $O(n \log n)$ time using FFT. Due to space constraints, details will be given in the full version of the paper.

We consider a shifted binary alphabet $\Sigma' = \{1, 2\}$ and identify the wildcard $\star$ with the number 0. Obviously, one needs to replace in the protocol from **Fig. 2** the protocol π_{isBit} by π_{isTrit}. In order to compute (3) via FFT, it is also necessary for Alice to provide encryption of ${t_i}^2$ and ${t_i}^3$ for $i \in [\![n]\!]$ and for Bob to provide encryption of ${p_j}^2$ and ${p_j}^3$ for $j \in [\![m]\!]$. They have to prove the consistency of these ciphertexts with $\boldsymbol{a_i}$ for $i \in [\![n]\!]$ and $\boldsymbol{b_j}$ for $j \in [\![m]\!]$ using π_{eqDL}. The algorithm then follows the protocol from **Fig. 3** in order to get the Hamming distance (with wildcards) between $\mathcal{P}$ and $\mathcal{T}_i$ for $i \in [\![n-m+1]\!]$. We finally use the technique from **Fig. 1** to only reveal to Bob whether this distance is equal to 0 or not.

We can readily prove the security of this protocol (details will be given in the full version of the paper):

Theorem 4. *If π_{eqDL}, π_{Mult}, π_{Perm}, π_{nze} and π_{isTrit} are zero-knowledge proofs of knowledge secure against malicious verifiers for the languages $\mathcal{R}_{\text{eqDL}}$, $\mathcal{R}_{\text{Mult}}$,*

$\mathcal{R}_{\mathrm{Perm}}$, $\mathcal{R}_{\mathrm{nze}}$ and $\mathcal{R}_{\mathrm{isTrit}}$, if π_{KeyGen} and π_{Dec} are protocols secure against malicious verifiers for the functionality $\mathcal{F}^{\mathrm{KeyGen}}$ and $\mathcal{F}^{\mathrm{Dec}}$ and if the ElGamal encryption scheme is semantically secure, then the protocol outlined above securely computes $\mathcal{F}_{\mathrm{PM}\star}$ in the presence of malicious adversaries.

6 Conclusion

We presented protocols for secure two-party computation of generalized pattern matching. Our schemes can easily be combined in order to solve, for instance, the approximate pattern matching with wildcards problem. They can be extended in various directions: they can handle larger alphabets, longer pattern, and variants where the length of the pattern or the text remains hidden (as the schemes from [17]). Our technique can also provide round-optimal protocols (with similar efficiency) with universally composable security in the common reference string model (using the Groth-Sahai proof system [14]). Details will be given in the full version of the paper.

Acknowledgements. The author would like to thank Benoît Libert and anonymous referees for their comments and Carmit Hazay for helpful discussions on security models. The author was supported in part by the European Commission through the ICT Program under contract ICT-2007-216676 ECRYPT II.

References

1. M. Abe, R. Cramer & S. Fehr – "Non-interactive distributed-verifier proofs and proving relations among commitments", *Advances in Cryptology* – Asiacrypt *2002* (Y. Zheng, ed.), Lect. Notes Comput. Sci., vol. 2501, Springer, 2002, p. 206–223.
2. K. R. Abrahamson – "Generalized String Matching", *SIAM J. Comput.* **16** (1987), no. 6, p. 1039–1051.
3. R. Canetti – "Security and composition of multiparty cryptographic protocols", *J. Cryptology* **13** (2000), no. 1, p. 143–202.
4. D. Chaum & T. P. Pedersen – "Wallet Databases with Observers", *Advances in Cryptology* – Crypto *'92* (E. F. Brickell, ed.), Lect. Notes Comput. Sci., vol. 740, Springer, 1993, p. 89–105.
5. J. H. Cheon, S. Jarecki & J. H. Seo – "Multi-Party Privacy-Preserving Set Intersection with Quasi-Linear Complexity", IACR ePrint Archive, Report 2010/512, 2010.
6. P. Clifford & R. Clifford – "Simple Deterministic Wildcard Matching", *Inf. Process. Lett.* **101** (2007), no. 2, p. 53–54.
7. J. W. Cooley & J. W. Tukey – "An algorithm for the machine calculation of complex fourier series", *Math. Comp.* **19** (1965), p. 297–301.
8. R. Cramer, R. Gennaro & B. Schoenmakers – "A Secure and Optimally Efficient Multi-Authority Election Scheme", *Advances in Cryptology* – Eurocrypt *'97* (W. Fumy, ed.), Lect. Notes Comput. Sci., vol. 1233, Springer, 1997, p. 103–118.
9. M. Crochemore & W. Rytter – *Jewels of Stringology*, World Scientific Publishing, Hong-Kong, 2002, 310 pages.

10. T. ElGamal – "A Public Key Cryptosystem and a Signature Scheme based on Discrete Logarithms.", *IEEE Trans. Inf. Theory* **31** (1985), p. 469–472.
11. M. Fischer & M. Paterson – "String Matching and Other Products", *Complexity of Computation* (R. Karp, ed.), SIAM-AMS, vol. 7, American Mathematical Society, 1974, p. 113–125.
12. R. Gennaro, C. Hazay & J. S. Sorensen – "Text Search Protocols with Simulation Based Security", *PKC 2010: 13th International Conference on Theory and Practice of Public Key Cryptography* (P. Q. Nguyen & D. Pointcheval, eds.), Lect. Notes Comput. Sci., vol. 6056, Springer, 2010, p. 332–350.
13. J. Groth – "A Verifiable Secret Shuffle of Homomorphic Encryptions", *PKC 2003: 6th International Workshop on Theory and Practice in Public Key Cryptography* (Y. Desmedt, ed.), Lect. Notes Comput. Sci., vol. 2567, Springer, 2003, p. 145–160.
14. J. Groth & A. Sahai – "Efficient Non-interactive Proof Systems for Bilinear Groups", *Advances in Cryptology* – Eurocrypt *2008* (N. P. Smart, ed.), Lect. Notes Comput. Sci., vol. 4965, Springer, 2008, p. 415–432.
15. C. Hazay & Y. Lindell – "Efficient Protocols for Set Intersection and Pattern Matching with Security Against Malicious and Covert Adversaries", *J. Cryptology* **23** (2010), no. 3, p. 422–456.
16. —, *Efficient Secure Two-Party Protocols*, Information Security and Cryptography, Springer-Verlag, 2010.
17. C. Hazay & T. Toft – "Computationally Secure Pattern Matching in the Presence of Malicious Adversaries", *Advances in Cryptology* - Asiacrypt 2010 (M. Abe, ed.), Lect. Notes Comput. Sci., vol. 6477, Springer, 2010, p. 195–212.
18. A. Jarrous & B. Pinkas – "Secure Hamming Distance Based Computation and Its Applications", *ACNS 09: 7th International Conference on Applied Cryptography and Network Security* (M. Abdalla, D. Pointcheval, P.-A. Fouque & D. Vergnaud, eds.), Lect. Notes Comput. Sci., vol. 5536, Springer, 2009, p. 107–124.
19. D. E. Knuth, J. H. Morris Jr & V. R. Pratt – "Fast Pattern Matching in Strings", *SIAM J. Comput.* **6** (1977), no. 2, p. 323–350.
20. M. S. Rahman & C. S. Iliopoulos – "Pattern Matching Algorithms with Don't Cares", *SOFSEM 2007: Theory and Practice of Computer Science* (J. van Leeuwen, G. F. Italiano, W. van der Hoek, C. Meinel, H. Sack, F. Plasil & M. Bieliková, eds.), Institute of Computer Science AS CR, Prague, 2007, p. 116–126.
21. C.-P. Schnorr – "Efficient Signature Generation by Smart Cards", *J. Cryptology* **4** (1991), no. 3, p. 161–174.
22. T. Schoenmeyr & D. Y. Zhang – "FFT-based algorithms for the string matching with mismatches problem", *J. Algorithms* **57** (2005), no. 2, p. 130–139.
23. J. R. Troncoso-Pastoriza, S. Katzenbeisser & M. Celik – "Privacy preserving error resilient dna searching through oblivious automata", *ACM CCS 07: 14th Conference on Computer and Communications Security* (P. Ning, S. D. C. di Vimercati & P. F. Syverson, eds.), ACM Press, 2007, p. 519–528.
24. Y. Tsiounis & M. Yung – "On the Security of ElGamal Based Encryption", *PKC'98: 1st International Workshop on Theory and Practice in Public Key Cryptography* (H. Imai & Y. Zheng, eds.), Lect. Notes Comput. Sci., vol. 1431, Springer, 1998, p. 117–134.
25. A. C. Yao – "Protocols for Secure Computations", *23rd Annual Symposium on Foundations of Computer Science*, IEEE Computer Society Press, 1982, p. 160–164.