Efficient and secure computation of edit distance on genomic data

Andrea Migliore
Department of Computer Science, Università degli Studi di Milano
Stelvio Cimato (stelvio.cimato@unimi.it)
Department of Computer Science, Università degli Studi di Milano
Gabriella Trucco
Department of Computer Science, Università degli Studi di Milano

Research Article

Keywords: Secure multi-party computation, Garbled circuit, Edit distance, Computational performance

Posted Date: November 24th, 2022

DOI: https://doi.org/10.21203/rs.3.rs-2176027/v1

Additional Declarations: No competing interests reported.

RESEARCH

Efficient and secure computation of edit distance on genomic data

Andrea Migliore, Stelvio Cimato* and Gabriella Trucco

Abstract

Background: Genetic information are the most sensitive data for a person and must be protected from malicious attacks. In this paper we focus on the application of secure multi-party computation, a subfield of cryptography, to the computation of edit distance, one of the most used metrics among genetic similarity indicators, useful for the diagnosis and treatment of many genetically based diseases.

Results: We analyze four algorithms and compare them to the best prior results found in literature [1]: (1) the Wagner-Fischer algorithm [2], using the entire dynamic programming matrix; (2) the Wagner-Fischer algorithm, optimized to use only the minimum needed columns for the computation; (3) the Ukkonen algorithm [3], considering a threshold of approximately 60% of the longest string; (4) the Ukkonen algorithm, using a generalized cut-off technique which reduces the number of cells to be computed. The Ukkonen algorithm with generalized cut-off is the one that performed better among the considered algorithms and it also proved to have better performances than the best prior results found in literature.

Conclusions: Securely computing the edit distance between human genomes have become very important in medical and public health domains. Improving computational performance is a key factor for real-world application scenarios.
In this work, we proposed several secure implementations of some of the most efficient edit distance algorithms, achieving better performances over existing protocols found in literature [4]. Moreover, this is the first time the Ukkonen's algorithm is proposed using all possible state-of-the-art optimizations for garbled circuits. The algorithms and protocols used in this work were also applied on both random, high-entropy, and real genomic, low-entropy strings and are provably secure with respect to the standard definition of security for Multi-Party Computation (MPC) protocols.

Keywords: Secure multi-party computation; Garbled circuit; Edit distance; Computational performance

Background

The digital transformation is having a deep impact on many activities of our daily lives, creating both opportunities and challenges. Opportunities are given by the digitalization that has improved organisational performance by fostering innovation and boosting entrepreneurial initiatives giving the possibility to open new businesses. Challenges are given by the fact that sensible data and in general privacy of the users can be exposed to many security risks caused by the sharing and the elaboration of the associated information.

Bionformatics is a rapidly advancing field, where the application of information technology to the treatment of biological data is helping in better analyzing and understanding the various types of data resulting from different biological processes. Last years have registered a tremendous advance in speed and cost reduction, allowing the completion of the Human Genome Project and the possibility to sequence a full genome for a reduced quantity of money. However the intersection of genomics and security arises stimulating ethical and social issues that need to be addressed, since genetic information can be considered *the* most sensitive data for a person and consequently, it must be protected from any kind of malicious attack or disclosure. *Secure multi-party computation* (MPC) [5], is a branch of cryptography whose goal is to enable a group of independent data owners, who do not trust each other or any common third party, to jointly compute a function that depends on all of their private inputs. There are a

*Correspondence: stelvio.cimato@unimi.it
Department of Computer Science, Università degli studi di Milano, Milan, Italy
Full list of author information is available at the end of the article

number of efficient implementation of these protocols based on ad-hoc techniques, which are developed to solve specific problems (such as set intersection) or on the generic transformation of the computed function, such as in the case of *garbled circuits* protocol [6] [7]. Secure multi-party computation can be used to solve a wide variety of real-life problems where sensitive data may be compromised.

In this paper, we focus on the application of secure multi-party computation to bioinformatics and more specifically to the computation of the edit distance, generically used to compute the distance between two strings. The human genome is composed by two complementary strands, with 3 billion DNA bases each. Each unit consists of Adenine (A), Cytosine (C), Guanine (G), and Thymine (T), the nitrogenous bases that constitute nucleotides. In this case, *edit distance* is an important metric to quantify how dissimilar two DNA strings are, by counting the minimum number of operations required to transform one string into the other. It is one of the most used and well established metrics among genetic similarity indicators because it is very useful for the diagnosis and treatment of many genetically based diseases such as cancer, Alzheimer's, schizophrenia and others [8] [9] [10] [11].

However, consider, for example, the problem of comparing a person's DNA against a database of cancer patients' DNA, with the goal of finding if the person is in a high risk group for a certain type of cancer. Such a task has very important health and social benefits, but DNA information is highly sensitive and should not be revealed to anyone during the computation.

Recent works have shown that if large volume of data associated to patients are available, then they can be exposed to privacy breaches, just only analyzing the aggregate statistics. As a consequence, several published results have been removed from public databases, to prevent the identification of individuals [12] [13]. By using secure multi-party computation parties involved can be confident that no sensitive information will be disclosed during or after the computation, revealing only the category of cancer they are close to (or none). Nothing else is revealed, neither the DNA of the person being compared nor the DNA of the patients in the database. This type of computation also ensures an additional guarantee, i.e. a malicious party cannot change the final result, for example, by making the person believe they have cancer and therefore propose expensive treatments. However, securely computing this and other similar metrics is a highly challenging research task because there exist several properties that need to be met in order to run these secure protocols. The scientific community paid attentions on this problem for over a decade [14]; and, even if, initially, secure edit distance computation was particularly slow, now state-of-the-art solutions are getting closer to be applicable to real genome sequences. A lot of papers have proposed several approaches over the years, including exact algorithms, approximations and heuristics, focusing also on the analysis of their performances [1] [15] [16] [17] [18] [19]. Most of these works do not precisely define and provide all the details used for the experiment setup, such as machine type, implementation and benchmarking settings, source codes, used input data, custom optimizations, etc., making it difficult to have a framework available to have an efficient solution.

In this work, we propose some secure implementations of the most efficient edit distance algorithms, achieving better performances over existing protocols found in literature, without sacrificing security and accuracy. Moreover, a secure implementation of the *Ukkonen*'s algorithm with the generalized cut-off technique using all possible state-of-the-art optimizations for garbled circuits is proposed. Also, the proposed algorithms are applied on both random, high-entropy, and real genomics, low-entropy strings and are provably secure with respect to the standard definition of security for MPC protocols. Finally, the methodology, all the experiment setup details, and the source codes used for this study are described and provided, defining a clear baseline for future works and enabling anyone to reproduce the experiments independently.

The paper is organized as follows: first we introduce the basic notions about edit distance and secure multi-party computation. In the "Methods" section, we describe the optimized techniques for the computation of the edit distance, and their implementation in SMPC. Finally we report the basic results and draw conclusions.

Preliminaries

In this section we report the basic notions about the two main aspects considered in this study: the computation of the edit distance and the technique to compute securely a function among collaborating parties.

Edit distance

Given two strings s_1 and s_2 on an alphabet Σ, the *edit distance* $d(s_1, s_2)$ is the minimum number of edit operations required to transform s_1 into s_2 and measures the similarity between two string sequences. An edit operation is the basic step in transforming a word into another word. There are different types of edit distance that allow different combinations of editing operations. In computational biology, one of the most frequently used type of edit distance is the **Levenshtein edit distance** [20].

In more details, a word over the finite alphabet Σ is a sequence $a_i, ..., a_n$ of symbols where $a_i \in \Sigma$ for $i = 1, 2, ..., n$. The empty word is denoted by the symbol ϵ. An edit operation is a pair (a, b) with $a, b \in \Sigma \cup \{\epsilon\}$ and $ab \neq \epsilon$ and is a basic step in transforming a word into another word. There are three types of operations: insertion, deletion and substitution. The edit operation is called an insertion if $a = \epsilon$, a deletion if $b = \epsilon$, and a substitution if $a \neq \epsilon \neq b$.
Each of these operations is associated with a cost. Usually, the cost $c(a \to b)$ for $a \neq b$ is 1, whereas for $a = b$ is 0. This results in a cost of 1 in the case of insertion, deletion or replace and 0 in the case where the two letters are the same.
Therefore, an edit sequence S is a sequence of edit operations $S = ((a_1, b_1), ..., (a_n, b_n)), n \geq 1$.
The final cost of an edit sequence S is defined as $C(S) = \sum_{i=1}^{n} c(a_i, b_i)$ [17].

Over the years, several exact algorithms, which seek the optimal answer to the problem, without considering margins of error or specific conditions have been proposed. In 2014 it has been proved [21] that the Levenshtein distance of two strings of length n cannot be computed in time $O(n^{2-\varepsilon})$ for any ε greater than zero unless the strong exponential time hypothesis (SETH) is false. This issue introduces strong limitations on the research for exact optimization of edit distance because, despite $O(n^2)$ is a polynomial time, it is not acceptable for specific applications or for very long strings.

The most famous exact algorithm is the *Wagner-Fischer algorithm* [2], based on dynamic programming that has a time complexity of $O(nm)$ where n and m are the string lengths to compare.
Since exact algorithms require at least $O(nm)$ operations to compute the Levenshtein distance, over the years, scientific research has been strongly committed to finding better alternatives through approximate algorithms that are generally faster than exact ones [22]. Typically, approximation algorithms search for approximate matches of a pattern from a string using also a predetermined maximum error threshold that indicates the maximum allowed edit distance for an approximate match. The most important algorithm in this category is the *Ukkonen's algorithm* [3] that optimizes the computation of edit distance by trying to restrict the number of cells that must be filled in the dynamic programming table.
If we are not interested in an edit distance greater than some maximum threshold k, then it is necessary to calculate only the diagonal band of width $2k + 1$ of the matrix because the other cells are irrelevant for the computation.
This algorithm achieves a time complexity of $O(k \times min(m, n))$, which is a substantial improvement in performance.
An advantageous optimization of the Ukkonen's algorithm with a generalization of the *cut-off* technique, described and used in the final experiment, leverages an implicit upper bound of the Levenshtein distance and can be used without a pre-specified threshold parameter.
This optimization reduces by approximately $(\frac{n}{2} + 1)(\frac{n}{2} + 2)$ the number of cells to be computed, resulting in a substantial computational gain.
Other important algorithms are also *Enhanced Ukkonen* [23], *Myers' Bit-Vector* [24], and *Wu algorithm* [25].

Secure Multi-Party Computation

Secure multi-party computation, introduced by Andrew Yao in the early 1980s [5], can be defined as the problem of n players who want to compute an agreed function of their inputs in a secure way. Formally, we assume $x_1, \ldots, x_n$ inputs, where player i knows x_i, and we want to compute $f(x_1, \ldots, x_n) = (y_1, \ldots, y_n)$ such that player i is guaranteed to learn y_i and nothing more than that.
At the beginning secure computation was only a theoretical interest but in the 2000s, algorithmic improvements and computing costs reached a point where it became realistic to think about building practical systems using general-purpose multi-party computation.
The first project that implemented this type of system was Fairplay [26]. For the first time, with Fairplay, it was possible to express a privacy-preserving program in a high level language and compile it to executables that could be run by the parties involved in the computation. However, its scalability and performance limited its use to toy programs.
The speed of the MPC protocol significantly increased over time and today is several orders of magnitude higher due to a combination of cryptographic, protocol, network and hardware improvements. These improvements made possible the adoption of MPC implementations in important contexts and applications [27].
Thanks to the increasingly efficient protocols for MPC that have been proposed in recent years, MPC can now be considered as a practical solution to several real-life problems.

Yao's Garbled Circuits Protocol

Yao's Garbled Circuits protocol (GC) is the most widely known MPC technique [6] [7].
Many MPC protocols are built on Yao's GC. Furthermore, Yao's GC runs in constant rounds and avoids

the costly latency associated with approaches where the number of communication rounds scales with the circuit depth.
The starting point for this and all other protocols is the same: we want to evaluate a given function $F(x, y)$ where party P_1 holds $x \in X$ and P_2 holds $y \in Y$.
One toy problem, often used in these context to better understand the protocol, is the *Yao's Millionaires' problem* [5]. It is about two millionaires, Alice and Bob, who are interested in knowing which of them is richer without revealing their actual wealth. The following description of the Yao's protocol is inspired by [28].
First of all we have to convert the function we need into a boolean circuit.
Then we have to evaluate each gate securely. For doing this Alice picks two random keys for each wire (inputs and outputs). One key corresponds to 0, the other to 1. There are a total of 6 keys for a 2-input gate, as shown in Figure 1.

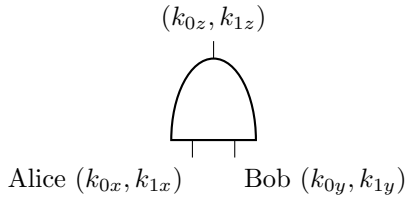

Figure 1 Wires for an AND Gate

Alice encrypts each row of the truth table by encrypting the output key with the corresponding pair of input keys.
Alice randomly permutes ("garbles") the encrypted truth table and sends it to Bob.

$$\text{Garbled truth table:} \quad \begin{matrix} E_{k_{1x}}(E_{k_{0y}}(k_{0z})) \\ E_{k_{0x}}(E_{k_{0y}}(k_{0z})) \\ E_{k_{1x}}(E_{k_{1y}}(k_{1z})) \\ E_{k_{0x}}(E_{k_{1y}}(k_{0z})) \end{matrix}$$

Note that Bob doesn't know which row of garbled table corresponds to which row of original table.
Then Alice sends to Bob the corresponding key to her input bit. Since keys are random, Bob won't know what this bit is.
The next step consists in running the oblivious transfer protocol between the two keys k_{0y}, k_{1y} and Bob's 1-bit input in order to give the correct key to Bob.
Bob now holds two keys and is able to decrypt exactly one of the output wire keys.
Proceeding with this method for all gates Bob evaluates the whole garbled circuit and sends the final result to Alice.
Neither Alice nor Bob learn any more information than what they know.

Methods

The human genome is composed by two complementary strands, with 3 billion DNA bases each. Each unit consists of Adenine (A), Cytosine (C), Guanine (G), and Thymine (T), the nitrogenous bases that constitute nucleotides. Between two randomly selected individuals, over 99% of their nucleotides are identical, just 1% of their DNA is due to genetic variations. *Single Nucleotide Polymorphism* (SNP) are the most common variations and involves only a single nucleotide. According to [29], about 50 million nucleotides in human genome are marked as SNPs, while two average individuals' genomes typically differ in 4-5 million variation sites.
Thus, studies that address this type of issues typically focus on SNP changes and therefore on quantifying the similarity of two DNA sequences.
We now provide a quick overview of the algorithms used for the computation of edit distance.

Wagner-Fischer algorithm

The Wagner-Fischer algorithm is based on dynamic programming and starts from the observation that it is possible to create a matrix with a number of rows and columns corresponding to the length of the two strings to be processed. For example, the two strings "GCTATAC" and "GCGTATGC" build the following matrix:

	ϵ	G	C	T	A	T	A	C
ϵ								
G								
C								
G								
T								
A								
T								
G								
C								

The final edit distance $D(n, m)$, where n and m represent the length of the two strings, is obtained by resolving all the edit distances of the substrings that constitute the final strings. In other words, the computation proceeds with each $D(i, j)$ where i and j are smaller values of n and m. The key idea is to solve all the sub-problems relying on the values obtained from the previous computation. Basically, we have to compute $D(i, j)$ for every $0 < i < n$ and $0 < j < m$. Each cell represents the edit distance of two substrings and, consequently, one of these sub-problems. Let X =

"GCGTATGC" and Y = "GCTATAC" be two strings and let α and β be the substrings consisting of all the characters of the sequence but the last element:

$$\underbrace{\text{GCGTATG}}_{\alpha}\text{C} \qquad \underbrace{\text{GCTATA}}_{\beta}\text{C}$$

we can define the edit distance between two strings as follows:

$$\text{EditDist}(\alpha x, \beta y) = min \begin{cases} \text{EditDist}(\alpha, \beta) + \delta(x, y) \\ \text{EditDist}(\alpha x, \beta) + 1 \\ \text{EditDist}(\alpha, \beta y) + 1 \end{cases}$$

where $\delta(x, y)$ is a value based on the difference between the two compared characters in that particular computation step. If $x = y$ then $\delta(x, y) = 0$, otherwise $\delta(x, y) = 1$.
Filling the entire matrix we obtain Table 1.

Table 1 Filled matrix

	ϵ	G	C	T	A	T	A	C
ϵ	0	1	2	3	4	5	6	7
G	1	0	1	2	3	4	5	6
C	2	1	0	1	2	3	4	5
G	3	2	1	1	2	3	4	5
T	4	3	2	1	2	2	3	4
A	5	4	3	2	1	2	2	3
T	6	5	4	3	2	1	2	3
G	7	6	5	4	3	2	2	3
C	8	7	6	5	4	3	3	**2**

Hence, the edit distance between the strings X and Y is 2.

This algorithm has a time complexity of $O(nm)$ where n and m are the string lengths to compare ($O(n^2)$ if $n = m$). This is because each value is dependent on the previous ones and to reach the final solution it is necessary to compute a table of $n \times m$ cells. Since every value is used to efficiently compute the final solution, the full dynamic programming table needs to be constructed, therefore we can easily conclude that the space complexity is equal to the time complexity.

If, however, we are not interested in reconstructing the optimal alignment between the two strings, it is possible to considerably reduce the space complexity by observing that, it is not necessary to construct the entire table but it is enough to fill the current row and overwrite the values at each step. Hence, space complexity drops to $O(min(m, n))$. This optimization does not have any impact on time complexity but, in practice, the algorithm should be faster than using the full matrix, because the CPU caching handles fewer values and therefore it does not need to manage large areas of memory.

Algorithm 1: Wagner-Fischer

```
n ← length(s1)
m ← length(s2)
for x ← 0 to m do
    M_{i,0} ← i
end
for y ← 0 to n do
    M_{0,y} ← y
end
for x ← 1 to m do
    for y ← 1 to n do
        r ← s(x − 1) = t(y − 1) ? 0 : 1
        M_{x,y} ← min(M_{x−1,y} + 1, M_{x,y−1} +
          1, M_{x−1,y−1} + r)
    end
end
return M[m][n]
```

Ukkonen's algorithm

Edit distance is not an NP-hard problem but in the 2014 it has been proved that the Levenshtein distance of two strings of length n cannot be computed in time $O(n^{2-\varepsilon})$ for any ε greater than zero unless the strong exponential time hypothesis (SETH) is false [21]. Obviously, this introduces strong limitations on the research for exact optimization of edit distance because, despite $O(n^2)$ is a polynomial time, it is not acceptable for specific applications or for very long strings.

In 1985 Esko Ukkonen proposed an algorithm [3] to optimize the computation of edit distance by trying to restrict the number of cells that must be filled in the dynamic programming table constructed in the Wagner-Fischer algorithm. Given m and n the two string lenghts, and i, j the coordinates of any cell in the matrix, from the diagonal and adjacency properties, Ukkonen observed that if $D(i, j) \leq k$ and $m \leq n$ then it is sufficient to fill only the cells in the diagonals $-\lfloor\frac{(k-n+m)}{2}\rfloor, -\lfloor\frac{(k-n+m)}{2}\rfloor + 1, \ldots, \lfloor\frac{(k+n-m)}{2}\rfloor$ of the dynamic programming matrix. He concluded that the $d(i, j)$ values form a non-decreasing sequence along any given diagonal, i.e.:

$$d(i, j) - 1 \leq d(i - 1, j - 1) \leq d(i, j)$$

consequently, it's necessary to calculate only the values that do not exceed the chosen threshold k.

Once $D(i, j) > k$, the cells $D(i + h, j + h)$ where $h \geq 0$ are irrelevant for computation purposes. This technique is called *cut-off* because, intuitively, it cuts out unnecessary values. For example, let's consider two random strings "ACAGACA" and "CAATCA", and a threshold $k = 2$. We obtain a Table like 2.

Table 2 Ukkonen matrix, k = 2

		A	C	A	G	A	C	A
C		1	1	2				
A		1	2	1	2			
A		2	2	2	2	2		
T			3	3	3	3	3	
C				4	4	4	3	4
A					4	4	4	3

We can see that the grey cells are strictly bigger than 2, hence we do not bother evaluate them. Therefore, the final edit distance is 3. For long strings this is a very high advantage because it is possible to significantly reduce the number of elements to be computed. We can find the edit distance by evaluating $n(2k+1)$ elements at most, where n is the length of the smaller string. This algorithm has a time complexity of $O(k \times min(m,n))$, which is a substantial improvement in performance.

Generalized Ukkonen's algorithm

The cut-off technique can be applied even without a pre-specified threshold parameter k [3] [30]. The Levenshtein distance does indeed have an implicit upper bound. Suppose we have two strings S and T whose lengths are n and m with $m \geq n$. We can guarantee that the Levenshtein distance cannot exceed $LevenshteinDist(S,T) \leq m$.
If we define the two strings as follows:

$$S = S_1 S_2 \dots S_{n-1} S_n$$

$$T = T_1 T_2 \dots T_{m-1} T_m$$

since $m \geq n$, we can rewrite:

$$S = S_1 S_2 \dots S_{n-1} S_n$$

$$T = T_1 T_2 \dots T_{n-1} T_n \dots T_m$$

Now we can convert T into S by replacing the sequence $T_1 \dots T_n$ with $S_1 \dots S_n$ and deleting $T_{n+1} \dots T_m$. The cost of performing this operation is at most m because it needs n substitution and $(m-n)$ deletions. Hence, the upper bound is proved. This upper bound, so that $LevenshteinDist(S,T) \leq m$, is quite useful because it allows us to optimize the number of cells to be computed. Let X_c be the Manhattan distance from a cell to the upper right corner. Then, as a rule, we can say that as long as the expression:

$$(n - X_c) + (m - X_c) \geq m \quad \text{con } X_c \geq 0$$

Table 3 Generalized Ukkonen

		A	C	G	T
	0	1			
G	1	1	2		
A		1	2	3	
C			1	2	3
T				2	2

is valid, we can safely ignore the corresponding matrix cells. A simple example is given in Table 3.
Assuming two strings have the same lenghts $n = m$, then:

$$(n - X_c) + (n - X_c) \geq n$$
$$2n - 2X_c \geq n$$
$$n - 2X_c \geq 0$$
$$n \geq 2X_c$$
$$X_c \leq \frac{n}{2}$$

The ignored cells in one side of the matrix (upper right) and in the opposite side (bottom left) have an arithmetic progression of $(X_c + 1)(X_c + 2)$, because if $X_c = 0$ we ignore 2 cells, if $X_c = 1$ we ignore 6 cells, and so on. Thus, this optimization reduces by approximately

$$\left(\frac{n}{2} + 1\right)\left(\frac{n}{2} + 2\right)$$

the number of cells to be computed.

Securely computing edit distance

Edit distance, weighted edit distance and Needleman-Wunsch algorithms are often used and widely adopted in the bioinformatics research field. However, securely computing these metrics is a highly challenging research task. Over the last years, researchers have intensively studied several ways to meet the vital properties needed to run these secure protocols.

Many research works focused on secure and efficient implementations of edit distance algorithms and the analysis of their benchmarks [1] [15] [16] [17] [18] [19]. However, almost none of these works precisely define and provide all the details used for the experiment setup, such as machine type, framework chosen, implementation and benchmarking settings, source codes, used input data, custom optimizations, etc. In this work we discuss novel secure implementations of existing edit distance algorithms using the current state-of-the-art garbled circuits, analyze their performance, and define a clear baseline for future works.

In order to provide meaningful and reproducible results, we decided to use Google Cloud Platform. For

Algorithm 2: Generalized Ukkonen's

```
n ← length(s1)
m ← length(s2)
gmax ← (m − 1)/2
gmin ← 1 − gmax − (n − m)
for j ← 0 to gmax do
    row[j] ← j
end
for i ← 1 to n do
    row[0] ← i − 1
    min ← Max(gmin, 1)
    max ← Min(gmax, m)
    gmin ← gmin + 1
    gmax ← gmax + 1
    dia ← row[min − 1]
    top ← row[min]
    if s1[i − 1] ≠ s2[min − 1] then
        dia ← Min(dia, top) + 1
    end
    row[min] ← dia
    left ← dia
    dia ← top
    for j ← min + 1 to max do
        top ← row[j]
        if s1[i − 1] ≠ s2[j − 1] then
            dia ← Min(dia, top, left) + 1
        end
        row[j] ← dia
        left ← dia
        dia ← top
    end
    if m = max then
        continue
    end
    if s1[i − 1] ≠ s2[max] then
        dia ← Min(dia, left) + 1
    end
    row[max + 1] ← dia
end
dia ← row[m]
return dia
```

this work, the N1, a Compute Engine's first generation general-purpose machine type, was chosen; specifically the `n1-standard-1` instance, provided with 1 vCPU and 3.75 GB of memory and Ubuntu 21.04. To develop all secure implementations we chose the EMP-toolkit framework [31], because it integrates all existing applicable optimizations for garbled circuits including efficient OT extension [32] [33], FreeXOR technique [34] and Half-Gates garbling [35]. EMP-toolkit also provides a 127-bit computational security (κ).
For the purpose of this work a semi-honest model and two-party computation was chosen.
All algorithms were developed in C/C++.

In computer science field there is no common consensus regarding benchmark measurement. Many variables can distort the effective result of a process execution, mainly due to I/O operations, task switches, time spent on other processes, interrupt handling, etc. running in the same time span.
Generally, two methods are taken into account to measure how much time has passed: *Wall time* and *CPU time*.
For the purposes of this work it was decided to mainly adopt the CPU time, using the `clock_gettime` function with the `CLOCK_PROCESS_CPUTIME_ID` option, available in the GNU C Library. However some comparisons have been carried out using also the wall time.
Finally, to obtain a significant value, each test has been executed 10 times, and an arithmetic mean among all the results has been calculated.

Results

In this section, we analyze the performance of four privacy-preserving algorithms:

- **Wagner-Fischer (full matrix)**: the Wagner-Fischer algorithm, using the entire dynamic programming matrix.
- **Wagner-Fischer (two columns)**: the Wagner-Fischer algorithm, optimized to use only the minimum needed columns for the computation.
- **Ukkonen (with threshold)**: the Ukkonen's algorithm, considering a threshold of approximately 60% of the longest string.
- **Ukkonen (generalized cut-off)**: the Ukkonen's algorithm, using the generalized cut-off technique.

The source code is available publicly at GitHub [4] for interested readers.
The algorithms were evaluated using two datasets of DNA data.
The first dataset consists of randomly generated strings of various lengths of {A, C, G, T} elements. The lengths considered were: 200, 1000, 2000, 3000, and 4000. The second dataset consists of real DNA strings from a genome database released by the "iDASH Security and Privacy Workshop 2016" [36]. The database includes 50 strings of approximately 3400-3500 characters each.
Results are shown in table 4. Computation inputs considered are two n-nucleotide genomes. The threshold for Ukkonen algorithm is automatically computed

	String Lengths							
	Randomly Generated					Genomic (iDash)		
Algorithm	**200**	**1000**	**2000**	**3000**	**4000**	**3456**	**3465**	**3475**
Wagner-Fischer	0.2	8.1	34.9	-	-	-	-	-
Wagner-Fischer (opt.)	0.2	7.8	34.0	82.6	146.7	109.2	108.6	108.6
Ukkonen (thr.)	0.1	5.0	21.9	52.7	93.7	71.0	70.4	70.9
Ukkonen (gen.)	0.1	4.8	20.9	50.6	90.2	67.4	67.7	68.6

Table 4 Algorithms performance on different string lengths. Computation inputs are two n-nucleotide genomes. The threshold for Ukkonen algorithm is automatically computed as 60% of the longest string. The symbol "-" means that the algorithm was terminated by Linux `Out of Memory Killer` process before it ended. Times are in seconds.

as 60% of the longest string. The symbol "-" means that the algorithm was terminated by Linux `Out of Memory Killer` process before it ended.
In Table 5 we compared the results obtained in this work with the best prior results found in literature [1] (without considering particular and specific customization to the garbling scheme), revealing a significant increase in performance.

Table 5 Performance comparison. Computation inputs are two 4000-nucleotide genomes.

	CPU time (s)	Wall time (s)
Best Prior	N/A	286
This Work	**90.2**	**183**

For fair comparison with the cited work, the wall time has also been calculated, using the `CLOCK_REALTIME` option of the `clockgettime` function.
Hence, a 36% speedup was achieved by this work.
Both these works use high-entropy strings (randomly generated), are generic, accurate, provably secure under the standard, preferred definition of security, and have comparable benchmarking methodologies. However, this work does not benefit from any custom circuits optimization, present instead in the cited work and in many other works.

Conclusion

The computation of the edit distance between human genomes has become a very important task in medical domain.
In this paper we have proposed some novel techniques to securely compute the edit distance on human genomes and proposed an efficient implementation reporting some experimental results on both artificial and public datasets. Our techniques show improved efficiency over state of the art, reducing the overall time needed. Some more optimizations, on both the computation of the distance and the usage of secure computation framework can be developed.

Acknowledgements
Not applicable.

Funding
Not applicable.

Abbreviations
Not applicable.

Availability of data and materials
https://github.com/AndreaMigliore/smpc-ed

Ethics approval and consent to participate
Not applicable.

Competing interests
The authors declare that they have no competing interests.

Consent for publication
Not applicable.

Authors' contributions
All authors have contributed equally.

Authors' information
Department of Computer Science, Università degli Studi di Milano, Milan, Italy.

Author details
Department of Computer Science, Università degli studi di Milano, Milan, Italy.

References

1. Zhu, R., Huang, Y.: Efficient and precise secure generalized edit distance and beyond. IEEE Transactions on Dependable and Secure Computing (2020)
2. Wagner, R.A., Fischer, M.J.: The string-to-string correction problem. Journal of the ACM (JACM) **21**(1), 168–173 (1974)
3. Ukkonen, E.: Algorithms for approximate string matching. (1985)
4. Migliore, A.: smpc-ed (2021). https://github.com/AndreaMigliore/smpc-ed
5. Yao, A.C.: Protocols for secure computations. In: 23rd Annual Symposium on Foundations of Computer Science (sfcs 1982), pp. 160–164 (1982). IEEE
6. Yao, A.C.: How to generate and exchange secrets. In: 27th Annual Symposium on Foundations of Computer Science (sfcs 1986), pp. 162–167 (1986). IEEE
7. Micali, S., Goldreich, O., Wigderson, A.: How to play any mental game. In: Proceedings of the Nineteenth ACM Symp. on Theory of Computing, STOC, pp. 218–229 (1987). ACM

8. Koboldt, D.C., Fulton, R., McLellan, M., Schmidt, H., Kalicki-Veizer, J., McMichael, J., Fulton, L., Dooling, D., Ding, L., Mardis, E., *et al.*: Comprehensive molecular portraits of human breast tumours. Nature **490**(7418), 61–70 (2012)
9. Taylor, J.G., Choi, E.-H., Foster, C.B., Chanock, S.J.: Using genetic variation to study human disease. Trends in molecular medicine **7**(11), 507–512 (2001)
10. Waddell, N., Pajic, M., Patch, A.-M., Chang, D.K., Kassahn, K.S., Bailey, P., Johns, A.L., Miller, D., Nones, K., Quek, K., *et al.*: Whole genomes redefine the mutational landscape of pancreatic cancer. Nature **518**(7540), 495–501 (2015)
11. Evans, W.E., Relling, M.V.: Moving towards individualized medicine with pharmacogenomics. Nature **429**(6990), 464–468 (2004)
12. Homer, N., Szelinger, S., Redman, M., Duggan, D., Tembe, W., Muehling, J., Pearson, J.V., Stephan, D.A., Nelson, S.F., Craig, D.W.: Resolving individuals contributing trace amounts of dna to highly complex mixtures using high-density snp genotyping microarrays. PLoS genetics **4**(8), 1000167 (2008)
13. Wang, R., Li, Y.F., Wang, X., Tang, H., Zhou, X.: Learning your identity and disease from research papers: information leaks in genome wide association study. In: Proceedings of the 16th ACM Conference on Computer and Communications Security, pp. 534–544 (2009)
14. Atallah, M.J., Kerschbaum, F., Du, W.: Secure and private sequence comparisons. In: Proceedings of the 2003 ACM Workshop on Privacy in the Electronic Society, pp. 39–44 (2003)
15. Zhu, R., Huang, Y.: Efficient privacy-preserving general edit distance and beyond. Cryptology ePrint Archive (2017)
16. Kaghazgaran, P.: Privacy-preserving edit distance on genomic data. arXiv preprint arXiv:1711.06234 (2017)
17. Aziz, M.M.A., Alhadidi, D., Mohammed, N.: Secure approximation of edit distance on genomic data. BMC medical genomics **10**(2), 55–67 (2017)
18. Wang, X.S., Huang, Y., Zhao, Y., Tang, H., Wang, X., Bu, D.: Efficient genome-wide, privacy-preserving similar patient query based on private edit distance. In: Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, pp. 492–503 (2015)
19. Huang, Y., Evans, D., Katz, J., Malka, L.: Faster secure {Two-Party} computation using garbled circuits. In: 20th USENIX Security Symposium (USENIX Security 11) (2011)
20. Levenshtein, V.I., *et al.*: Binary codes capable of correcting deletions, insertions, and reversals. In: Soviet Physics Doklady, vol. 10, pp. 707–710 (1966). Soviet Union
21. Backurs, A., Indyk, P.: Edit distance cannot be computed in strongly subquadratic time (unless seth is false). (2014)
22. Hyyrö, H.: Practical methods for approximate string matching. (2003)
23. Berghel, H., Roach, D.: An extension of ukkonen's enhanced dynamic programming asm algorithm. ACM Trans. Inf. Syst. **14**(1), 94–106 (1996)
24. Myers, G.: A fast bit-vector algorithm for approximate string matching based on dynamic programming. J. ACM **46**(3), 395–415 (1999)
25. Wu, S., Manber, U., Myers, G., Miller, W.: An o(np) sequence comparison algorithm. Information Processing Letters **35**(6), 317–323 (1990)
26. Malkhi, D., Nisan, N., Pinkas, B., Sella, Y.: Fairplay—a secure two-party computation system. In: Proceedings of the 13th Conference on USENIX Security Symposium - Volume 13. SSYM'04, p. 20. USENIX Association, USA (2004)
27. Evans, V. David Kolesnikov, M. Rosulek, M.: A Pragmatic Introduction to Secure Multi-Party Computation. NOW Publishers, ??? (2020)
28. Cimato, S., Ciriani, V., Moroni, M.: Esop synthesis for secure computation. In: Bergakademie, T.U. (ed.) 11th International Workshop on Boolean Problems (IWSBP14) (2014)
29. Sherry, T., Ward, M.-H., Kholodov, M., Baker, J., Phan, L., Smigielski, E., Sirotkin, K.: dbsnp: the ncbi database of genetic variation. Nucleic acids research (2001)
30. Seiji, F.: Can We Optimize the Wagner-Fischer Algorithm? (2019). https://ceptord.net/wagner-fischer/index.html
31. Wang, X., Malozemoff, A.J., Katz, J.: EMP Toolkit (2016). https://github.com/emp-toolkit
32. Kolesnikov, V., Kumaresan, R.: Improved ot extension for transferring short secrets. In: Advances in Cryptology (CRYPTO) (2013)
33. Ishai, Y., Kilian, J., Nissim, K., Petrank, E.: Extending oblivious transfers efficiently. In: International Cryptology Conference (CRYPTO) (2003)
34. Kolesnikov, V., Schneider, T.: Improved garbled circuit: Free xor gates and applications. In: Proceedings of the 35th International Colloquium on Automata, Languages and Programming, Part II. ICALP '08, pp. 486–498. Springer, Berlin, Heidelberg (2008)
35. Zahur, S., Rosulek, M., Evans, D.: Two halves make a whole - reducing data transfer in garbled circuits using half gates. In: EUROCRYPT, pp. 220–250 (2015)
36. Tang, H., Wang, X., Kuo, T.-T., Ohno-Machado, L., Jiang, X., Harmanci, A.O., Kim, M.: IDash privacy & security workshop (2016). http://www.humangenomeprivacy.org/2016/index.html