Toward A Practical Data Privacy Scheme for A Distributed Implementation of the Smith-Waterman Genome Sequence Comparison Algorithm*

Doug Szajda Michael Pohl Jason Owen Barry Lawson
University of Richmond
Richmond, Virginia
{dszajda, mike.pohl, wowen, blawson}@richmond.edu

Abstract

Volunteer distributed computations utilize spare processor cycles of personal computers that are connected to the Internet. The resulting platforms provide computational power previously available only through the use of expensive clusters or supercomputers. However, distributed computations running in untrustworthy environments raise a number of security concerns, including computation integrity and data privacy.

This paper introduces a strategy for enhancing data privacy in some distributed volunteer computations, providing an important first step toward a general data privacy solution for these computations. The strategy is used to provide enhanced data privacy for the Smith-Waterman local nucleotide sequence comparison algorithm. Our modified Smith-Waterman algorithm provides reasonable performance, identifying most, and in many cases all, sequence pairs that exhibit statistically significant similarity according to the unmodified algorithm, with reasonable levels of false positives. Moreover the modified algorithm achieves a net decrease in execution time, with no increase in memory requirements. Most importantly, our scheme represents an important first step toward providing data privacy for a practical and important real-world algorithm.

Keywords: distributed computation, data privacy, Smith-Waterman algorithm

1. Introduction

Distributed volunteer computing platforms, in which personal computers connected to the Internet volunteer idle processor cycles to a large-scale distributed computation, enable computations once feasible only via expensive clusters or supercomputers. The computing power harnessed by these systems can top several petaflops, making them well suited for solving some SIMD-style parallel computations. Application domains benefiting from this technique include DNA gene sequence comparisons and protein folding in the biotechnology industry, advanced graphics rendering in the entertainment industry, exhaustive regression and other statistical applications in the financial industry, some forms of data mining, and Monte Carlo simulations. The typical computation in this setting is easily divisible into independent tasks small enough to be handled in a few hours by a typical personal computer.

In the common scenario, the *supervisor* of a volunteer distributed computing platform recruits *participants* who agree to allow the supervisor to execute code on their personal computers, either in exchange for some form of remuneration (in a commercial setting) or on a voluntary basis. Participants then download code that serves as the local execution environment for assigned computational *tasks*. For a given computation, participants are chosen, tasks are assigned and transmitted, and, as tasks are completed, significant results are returned to the supervisor. Though task results may be related, the tasks themselves are independent, so no communication is necessary between participants.

Because code is executed in untrustworthy environments, several security concerns are raised. Among the major issues is data privacy. Firms may have obtained data at great expense and are often reluctant to expose this proprietary information to unknown individuals. This is especially relevant in the biotechnology industry, where genetic data gleaned from years of experimentation is often closely guarded.

While there is a small but growing body of literature dealing with providing greater assurance of the validity of results of volunteer computations ([21, 22, 25, 32, 34, 35]), to our knowledge no research has addressed the associated issue of data privacy. There is a long history of research

*This work was partially supported by the National Science Foundation under grant IIS-0524239.

concerning computing with encrypted data ([2, 3, 5, 12, 15, 17, 29, 31]) that has resulted in many interesting and elegant results. Unfortunately, there are few real-world applications for which these methods are practical.

Our approach to providing data privacy is similar in spirit to computing with encrypted data. In that scenario, Alice has a function f and an input x. She wants Bob to compute $f(x)$ for her, but she does not wish to reveal x. So she encrypts x (creating $E(x)$), and asks Bob to compute $f'(E(x))$, where f' is such that Alice can easily determine $f(x)$ from $f'(E(x))$, but Bob cannot determine x from $E(x)$ and/or $f'(E(x))$. If this can be accomplished, f is said to be *encryptable*, and the transformations involved comprise an *encryption scheme* for f.

For a volunteer distributed computation that seeks to identify essential (in some sense) inputs the constraints are significantly relaxed. In this case, Alice has several x values and seeks to determine which are important in her context. Because Alice does not necessarily require the specific function values to make this determination, there is far greater flexibility in the definition of f'. We call this loosely defined notion *sufficient accuracy*.

Alice enjoys another advantage in the present context: she has several (possibly millions of) potential suitors offering computing services. Thus, the possibility exists that for each x, Alice can distribute the work of computing $f'(E(x))$ over several parties. Provided this can be done in a manner that ensures that no participant possesses sufficient information to determine x, then Alice can, in the absence of collusion, keep x confidential.

The notion of sufficient accuracy is particularly relevant for distributed optimization computations, i.e., those computations intended to locate optimum (or sufficiently close to optimum) values of some function f. These computations act as filters because identification of the distinguished input(s) is sufficient to determine the associated extreme value(s). Many important applications deployed on volunteer distributed computing platforms, including exhaustive regression, genetic sequence comparisons, and protein folding, are optimizing computations. In practice, the data identified as meaningful by participants are subsequently subjected to far more extensive, and typically more expensive, postprocessing analysis by the supervisor. This further relaxes problem constraints: data can sometimes be (mis)identified as important, provided the number of these *false positives* is sufficiently small, as long as truly important data is rarely, if ever, missed.

The specific contributions of this paper are to:

- Introduce the concept of *sufficient accuracy*.
- Present a strategy for enhancing data privacy in a practical and important real-world application: the Smith-Waterman local nucleotide sequence comparison algorithm. The importance of this algorithm is underscored by the fact that Smith-Waterman has been implemented in commercial distributed computing settings.
- Present a practical and important real-world application that requires data privacy and is efficiently parallelizable. Such applications have so far proven elusive. The present example thus represents a possible first entry for a benchmark suite of applications for privacy study.

The strategy described here is not applicable to *all* volunteer distributed optimization computations — there are cases in which the information required to determine the importance of data cannot be preserved without revealing the data itself. Nor do we claim that our scheme represents the final (or only) solution to the problem of data privacy for the Smith-Waterman algorithm. In many ways, our solution for this example is less than ideal. In the best possible situation, formal privacy and adversary models would be developed, and the efficacy of our solutions proved within that framework. Unfortunately, such models tend to be far too restrictive for real-world applications. Our approach, on the other hand, is more heuristic in nature. Heuristic solutions are problematic because they are not formally verified, and are thus often vulnerable to unanticipated attack strategies. Regardless, our solution represents a first step: for many configurations, our modified algorithm identifies all statistically significant sequences without a single false positive. Moreover its security can be reasonably estimated, though not rigorously measured, via entropy calculations. The second shortcoming of our strategy is that in no case, and for no specific configuration, do we achieve the sensitivity of the unmodified Smith-Waterman algorithm, which is theoretically guaranteed to find the best matching substrings from a pair of sequences. Our methods do, however, in many cases exhibit sensitivity comparable to Smith-Waterman. Finally, the methods here apply only to nucleotide sequence comparisons, not to amino acid sequence comparisons (for reasons that are discussed in Section 4).

The remainder of the paper is organized as follows. In Section 2 we present our model for the distributed computations and platforms under consideration. Section 3 discusses the general technique as applied to optimizing distributed volunteer computations. We provide brief introduction to biological sequence analysis in Section 4. Sections 5 and 6 present the details of our privacy scheme for Smith-Waterman and present related simulation results. Related work is discussed in Section 7. We present our conclusions in Section 8.

2. The model

We consider parallel computations in which the primary computation, the *job*, is easily divided into *tasks* small enough to be solved by a PC in a reasonable amount of time (typically on the order of several hours of CPU time). Individual tasks are independent of one another.

The computing platform consists of a trusted central control server or server hierarchy (which we denote using the term *supervisor*) coordinating typically between 10^4 and 10^7 personal computers in a master-slave relationship. These slave nodes, or *participants*, are assigned tasks by the supervisor. Participants download code, typically in the form of a screen saver or applet, that serves as the local execution environment for tasks. Because tasks are independent, communication between participants is unnecessary and, assuredly, not permitted.

Formally, a job consists of the evaluation of a function or algorithm $f : D \rightarrow R$ for every input value $x \in D$. Tasks are created by partitioning D into (possibly overlapping) subsets D_i, with the understanding that task $T(D_i)$ will evaluate $f(x)$ for every input $x \in D_i$. Each task $T(D_i)$ is assigned a filter function $G_i : P(R) \rightarrow P(\mathbb{Z}^+ \cup \{0\})$, where $P(R)$ is the power set of R. Each $x \in D$ is assumed to have a unique (nonnegative integer) identifier, $\mathrm{id}(x)$. The element $x \in D_i$ is considered significant (equivalently $f(x)$ is a significant result) if and only if $\mathrm{id}(x) \in G_i(f(D_i))$, where $f(D_i) \equiv \{f(x)|x \in D_i\}$. That is, the filter function returns the indices of significant elements $x \in D_i$. The filter functions have domain $P(R)$, rather than R itself, because the significance of an input may depend on its function value relative to the function values of other elements of D_i.

We assume the existence of a global intelligent adversary. The adversary possesses sufficient technical skills to efficiently decompile, analyze, and/or modify executable code as necessary. In particular, the adversary has knowledge both of the algorithm used for the computation and of the measures used to prevent data disclosure.

Attacks that result from a compromise of data in transit are beyond the scope of this paper — we assume the integrity of such data is verified by other means. In addition, we do not consider attacks that result from the compromise of the central server or other trusted management nodes. Finally, we do not consider attacks resulting from malicious participants returning incorrect results or through the collusion of such adversaries.

3. Leveraging Sufficient Accuracy

The theory behind the sufficient accuracy method is straightforward. The success of a task $T(D_i)$ in a filtering computation is based solely on whether important values in D_i are identified. Presumably the intrinsic value of input $x \in D_i$ will depend (at least in part) on the value $f(x)$. But determination of the importance of x, rather than returning the value $f(x)$, can sometimes be achieved using inputs x' and functions f' that differ significantly from x and f. In effect, considerable flexibility is introduced into the precise definitions of inputs and functions.

Our strategy for achieving data confidentiality with task $T(D_i)$ involves transforming the set D_i, function f, and filter function G into a set D'_i, function f' and filter function G' such that $T(D_i)$ can be replaced with the task $T(D'_i)$ consisting of the evaluation of f' on D'_i. Ideally, the transformation achieves the following ***transformation properties***.

1. The task $T(D'_i)$ should not leak any additional information about the values in D_i other than what can be learned from public sources and the values, $f(D_i)$, output by the untransformed function.

2. The set of identifiers $G'(T(D'_i))$ returned from $T(D'_i)$ contains the set of identifiers $G(T(D_i))$.

3. The difference $G'(T(D'_i)) - G(T(D_i))$ is reasonably small, where the definition of reasonable is application dependent.

In practice there is flexibility in these requirements. Some applications may tolerate a few missed important results provided that a certain proportion of identified important results are generated. Others may accept some flexibility on the number of false positives, provided that no important results are missed[1].

Note that the transformations here differ from traditional encryption algorithms and hash functions in important ways. Good encryption algorithms must be reversible, and should exhibit a strong avalanche effect. A strong avalanche effect in the current context, however, will likely obscure information to a degree that similar inputs will not be identified as such. Furthermore, our transformation, like hash functions, should not be reversible. However hash functions should also exhibit a strong avalanche effect. They must also be repeatable, while the transformations here need not be.

4. Smith-Waterman Sequence Comparison

A thorough treatment of sequence comparison techniques would (and does) fill several texts. This section gives a brief description of a dynamic programming alignment technique developed by Smith and Waterman [33].

[1]The notion that encryption/decryption schemes can have less than 100% accuracy is not unprecedented in the cryptographic literature (see e.g., [4], [6], [16])

The sequences that biologists study consist of either nucleotide bases (occurring in DNA fragments) or amino acids (the building blocks of proteins). We consider only DNA sequences, for which the underlying alphabet, Σ, consists of the set $\{A, C, T, G\}$ representing the nucleic acids adenine, cytosine, thymine, and guanine.

Let $U = u_1 u_2 \dots u_n$ be a sequence[2] over Σ. Sequences evolve primarily in three ways. Either an element of a sequence is removed (a *deletion*), an element is inserted (an *insertion*), or an existing element is transformed into a different element (a *substitution*). Biologists track evolutionary changes by writing the original sequence alongside the new sequence with appropriate positions aligned. For example, if U = CTGTTA, and u_2 undergoes a transformation from T to A, this would be written

$$\begin{array}{ll} U\colon & \text{CTGTTA} \\ V\colon & \text{CAGTTA} \end{array}$$

If instead u_4 is deleted from U, this is written

$$\begin{array}{ll} U\colon & \text{CTGTTA} \\ V\colon & \text{CTG–TA} \end{array}$$

where the '–' symbol acts as a placeholder, allowing the other symbols to remain aligned. Positions in a sequence with the '–' symbol are called *gaps*. If U is modified by inserting the nucleotide G in position 2 , this is represented by

$$\begin{array}{ll} U\colon & \text{C–TGTTA} \\ V\colon & \text{CGTGTTA} \end{array}$$

After several such mutations, U may have evolved significantly. We can represent this evolution with an alignment such as the following.

$$\begin{array}{ll} U\colon & \text{C–TGT––TA––} \\ V\colon & \text{CTA–TGCT–CG} \end{array}$$

In the example above, we assume that V evolves from U. In general, however, when given an alignment of two sequences, there is no implied origin — it is impossible to tell whether a particular gap is caused by a deletion or an insertion. Because of this symmetry, insertions and deletions are considered the same event, an *indel*.

Note that two sequences can be aligned in several ways, and that aligned sequences need not have the same length. Waterman [37] asserts that the number of alignments of two sequences of length n is asymptotically equal to $(2^{5/4}/\sqrt{\pi})(1+\sqrt{2})^{2n+1}(1/\sqrt{n})$. Thus, for example, two sequences of length 1000 have approximately 7.03×10^{763} distinct alignments.

Goodness of an alignment is measured using a scoring function s defined on pairs of symbols in Σ, and typically having the form

$$s(u,v) = \begin{cases} c & \text{if } u = v \\ -d & \text{if } u \neq v, \end{cases} \tag{1}$$

for nonnegative integers c and d with c equal to or slightly larger than d. This function can also be described using a matrix, called the *weight matrix*. The form of scoring function described by (1) is unique to nucleotide comparisons. Amino acid comparisons, in contrast, use scoring functions that do not exhibit this binary property (one score when symbols match, and one when they do not, regardless of the literal). Because our strategy for Smith-Waterman relies on this property, our methods are not applicable to amino acid comparisons.

Gaps are scored using a *gap function* (or *gap penalty*) g. Gap functions for local alignments typically have an *affine* form, with

$$g(k) = \alpha + \beta(k-1),$$

where k is the length of the gap[3], $\alpha > 0$ is the penalty of the initial indel in a multiple column gap, and $\beta > 0$ is the penalty for each subsequent indel in the gap. The score of an alignment is defined as the sum of the scores of each individual column, minus the gap penalties. By carefully choosing the scoring function and gap penalty, goodness of fit can be made to correspond to intuitive notions such as the probability that the sequences evolved from a common ancestor.

The similarity $S(U,V)$ of sequences U and V is defined to be the maximum score over all alignments between the two sequences. Although the number of alignments is huge, a dynamic programming algorithm developed by Needleman and Wunsch [26] allows the similarity of sequences of length n to be determined in $O(n^2)$ time.

The alignments discussed thus far are *global* alignments because they include every element of both sequences. In practice, *local alignments*, in which one seeks the best matching substrings of the two sequences, is much more useful. The local alignment problem seeks to find

$$\begin{aligned} &H(U,V) = \\ &\quad \max\{S(u_i u_{i+1} \dots u_{j-1} u_j, v_k v_{k+1} \dots v_{l-1} v_l) : \\ &\qquad 0 \le i \le j \le n-1,\ 0 \le k \le l \le m-1\}. \end{aligned}$$

Waterman [37] notes that even using Needleman-Wunsch (see Appendix A for details) for global alignments, the

[2] Though computer scientists typically begin sequences with index zero rather than one, biologists prefer to begin their sequences with index 1. We adhere to the biologists practice in this paper, so that the Smith-Waterman description here matches that in the biology literature. In addition, this convention eliminates the need for negative indices in the resulting dynamic programming matrices.

[3] The length of a gap is determined by the number of consecutive "dashes" in a *single* sequence. Thus, say, a column with a dash in sequence U, followed immediately by a column with a dash in sequence V is considered two length-one gaps, as opposed to a single length-two gap.

naive approach of computing the $\binom{n}{2}^2$ global alignments required for a single local alignment of two length-n sequences requires $O(n^6)$ time. Fortunately, the Smith-Waterman local alignment dynamic programming algorithm [33] reduces this to $O(n^3)$ time. Specifically, define for each (i, j) pair the function H by

$$H_{i,j} = \max\{0; S(u_x u_{x+1} \dots u_{i-1} u_i, v_y v_{y+1} \dots v_{j-1} v_j) : 1 \le x \le i,\ 1 \le y \le j\}.$$

Then H can be computed using the following two results.

Theorem. Assume that the gap function g is a function of gap length. Set $H_{0,0} = 0$, and set $H_{i,0} = H_{0,j} = 0$ for $1 \le i \le n$ and $1 \le j \le m$. Then

$$H_{i,j} = \max\left[0, \max_{1\le k\le i}\{H_{i-k,j} - g(k)\}, H_{i-1,j-1} + s(u_i, v_j), \max_{1\le l\le j}\{H_{i,j-l} - g(l)\}\right].$$

Corollary.

$$H(U, V) = \max\{H_{k,l} : 1 \le k \le n,\ 1 \le l \le m\}.$$

Significance of a Smith-Waterman score is based on probabilistic considerations. Specifically, similarity scores for random sequences compared using gapless (i.e., gaps are not allowed) Smith-Waterman were shown by Karlin and Altschul to follow an approximate extreme value distribution [23, 24]. Though their proof technique breaks down when gaps are allowed, empirical evidence ([7, 27]) has demonstrated that scores using gapped Smith-Waterman are approximately extreme value as well. The significance threshold value p is chosen so that a match will be considered significant provided the probability that a random comparison generates a score greater than or equal to p is small, typically less than 0.003. Note that p depends in part on the length of the sequences being compared, and thus varies between Smith-Waterman executions.

Tasks in a large-scale distributed implementation of Smith-Waterman consist of the comparison of two task-specific sets of sequences $\mathcal{A}$ and $\mathcal{B}$, with each sequence in one set compared to all sequences in the other. In most cases, one set consists of proprietary sequences, and the other consists of sequences contained in a public database such as the National Institutes of Health GenBank database [18]. In other cases, both sets consist of proprietary data. This would be the case, for example, if a company wished to use external participants to compare some newly identified sequences against the company's own large proprietary database. Unless otherwise specified, we assume for the remainder of this paper that sequences in $\mathcal{A}$ are proprietary and sequences in $\mathcal{B}$ are publicly available.

4.1 Sequence Comparison Assumptions

There is no uniform set of assumptions under which biologists run sequence comparisons. We consider here Smith-Waterman computations of an exploratory nature, in which the supervisor compares the proprietary sequences against sequences from a variety of species, in order to inform further small scale investigations. Chargaff [13] showed in 1951 that nucleotide frequencies are not uniform, but instead vary in known ways between species, and often vary among different evolutionary branches of the same species. Thus the total population of public sequences exhibits a wide range of nucleotide frequencies. Because we cannot know the exact distribution of the public database sequence population, we assume, both in our analysis and simulations, the worst case (in terms of determining sequence similarity) that all sequences in $\mathcal{A} \cup \mathcal{B}$ share the same relative nucleotide frequencies. We assume also, that nucleotide frequency distributions are reasonable, with, for example, all nucleotide frequencies between 0.15 and 0.35. With these assumptions, our algorithm performs in practice no worse (for the comparison configurations considered in this paper) than the results presented. It should be noted, however, that the security of our scheme *does* depend in part on the specific frequency distributions of the nucleotides being compared, and that there are pathological cases, such as a sequence consisting entirely of only one or two nucleotide literals, in which our proposed mechanisms provide little, if any, data privacy.

An actual large-scale distributed implementation of Smith-Waterman has tasks for which the sets $\mathcal{A}$ and $\mathcal{B}$ each contain approximately 100 sequences. We assume, without loss of generality and for relative simplicity of analysis, that $\mathcal{A}$ consists of a single sequence.

5. The Transformation

In applying the strategy outlined in Section 3 to the specifics of sequence comparison, our method for achieving data privacy requires transforming the sets $\mathcal{A}$ and $\mathcal{B}$, scoring and gap functions s and g, and filter parameter p into sets $\mathcal{A}'$, $\mathcal{B}'$, scoring functions s' and g' and filter parameter p'. We then assign the task $T(\mathcal{A}', \mathcal{B}', s', g', p')$ in place of $T(\mathcal{A}, \mathcal{B}, s, g, p)$.

Our transformation involves computing the offsets between occurrences of individual nucleotide literals. The resulting sequences of offsets are then distributed, and can be compared using the Smith-Waterman algorithm with the original scoring function and gap penalty information.

Specifically, for a sequence U over the alphabet Σ, and $\delta \in \Sigma$, let u_i^δ be the index (position) of the ith occurrence of δ in U. Define the *offset sequence*, $F(U, \delta)$ to be the

sequence

$$F(U,\delta) = \{u_1^\delta, u_2^\delta - u_1^\delta, u_3^\delta - u_2^\delta, \ldots, \ldots u_k^\delta - u_{k-1}^\delta\},$$

where the literal δ occurs in U exactly k times. Though we are assuming that all sequences begin at index 1, if for notational convenience we let $u_0^\delta = 0$, then we have

$$F(U,\delta)_i = u_i^\delta - u_{i-1}^\delta$$

for $i = 1, 2, \ldots, k$. For example, if U is the sequence

$$U: \quad \text{GCACTTACGCCCTTACGACG}$$

then the offset sequences for each $\delta \in \Sigma$ are

$$\begin{aligned} F(U,A) &= \{3,4,8,3\} \\ F(U,C) &= \{2,2,4,2,1,1,4,3\} \\ F(U,G) &= \{1,8,8,3\} \\ F(U,T) &= \{5,1,7,1\} \end{aligned}$$

For $\mathcal{E}$ a set of nucleotide sequences, let $F(\mathcal{E},\delta)$ denote the set $F(\mathcal{E},\delta) \equiv \{F(U,\delta) | U \in \mathcal{E}\}$. The basic scheme for transforming a sequence comparison task $T(\mathcal{A},\mathcal{B},s,g,p)$ is to randomly choose a nucleotide literal δ, compute $F(\mathcal{A},\delta)$ and $F(\mathcal{B},\delta)$ and send the task $T(F(\mathcal{A},\delta), F(\mathcal{B},\delta), s, g, p')$, where p' is a revised threshold. We determine the value of p' by applying statistical distribution fitting techniques to the results of small simulation runs. A detailed description of this is given below.

The intuition behind our method is that similar sequences should have similar offsets. Thus sequences with offsets that differ significantly from the sequence in $\mathcal{A}$ can be excluded. The security of the transformation, in contrast, results from its many-to-one property. Thus in order to avoid a black-box analysis (in which the adversary can reconstruct the sequence of nucleotides by identifying the substrings of a public sequence that match well with those of a proprietary sequence), the public databases should contain many sequences that could be the preimage, under our transformation, of the proprietary sequence. This, however, results in a potentially large number of false positives. These can be reduced significantly by creating two tasks, one corresponding to each of two nucleotide literals, and assigning those tasks to different participants. A sequence pair is then classified as significant only if *both* tasks indicate significant similarity.

We have found through experimentation that creating a set of tasks, each of which corresponds to a different nucleotide literal, greatly increases the accuracy of our scheme when compared with using only the offsets corresponding to a single nucleotide[4]. Two different methods, to be described next, for creating and using these multiple tasks performed well. In both methods, a single task from the unmodified computation becomes multiple tasks in the modified computation.

Maximum method: In the first method, all four offset sequences are computed for every sequence in the original task. Then four tasks are created, one for each nucleotide literal; each task contains the corresponding nucleotide offset sequence for each sequence in the original task. The four tasks measure significance against a single common threshold value, returning any matches that exceed the threshold. A pair of sequences is deemed well-matched (i.e., exhibit statistically significant similarity) provided the *maximum* of the four similarity scores exceeds the threshold. This approach can be augmented by requiring that two or more of the four similarity scores exceed the threshold, thus decreasing the rate of false positives.

Adding method: In the second method, the participants assigned each of the four tasks do not measure significance, but instead return all scores to a fifth participant, who adds the scores. Twp sequences are deemed well-matched provided the sum of the similarity scores for all four offset sequences exceeds the significance threshold. This approach, in which communication is required between participants and each participant generates significant network traffic, is not currently practical due to platform limitations. However, such network traffic will be far less an impediment in future platforms with increased bandwidth. Moreover, volunteer computing platforms in the near future likely will allow communication among participants via hierarchical architectures (as opposed to the flat master-slave architecture we have been considering). Such platforms have already been proposed [28] as a means of dealing with the inclusion of mobile wireless devices (such as cell phones and PDAs) into the computational grid.

Note that both schemes *decrease* net execution time as compared to the unmodified Smith-Waterman algorithm because they operate on shorter sequences. A Smith-Waterman comparison of two length-N sequences has time cost $O(n^3)$. Reducing sequence lengths by a factor of four (on average) decreases run time by roughly a factor of 64. Thus, even if five modified tasks are required to perform the work of a single unmodified task, one can expect an order of magnitude decrease in net execution time.

5.1 Analysis

We now consider the performance of our sequence transformation scheme as it relates to the ideal transformation properties described in Section 3.

Property 1 states that a transformed task $T(F(\mathcal{A},\delta), F(\mathcal{B},\delta), s, g, p')$ should not leak any information about the original private sequences $\mathcal{A}$ other

[4]This holds true as well for Needleman-Wunsch global sequence analysis. However in the global sequence alignment, using even a single offset provides high accuracy.

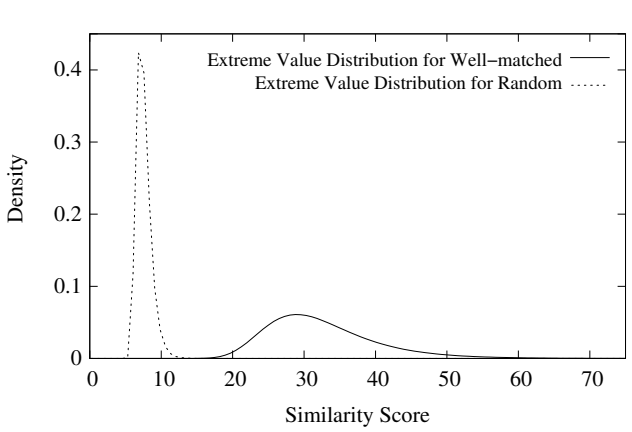

Figure 1. Distributions of Smith-Waterman scores using our transformation (with no mask) and the maximum method for determining significance. Curves generated from 10,000 comparisons with base sequence length between 600 and 800, matching portion length 300, and with well-matched sequences suffering an average of 52.5 substitutions and 52.5 indels.

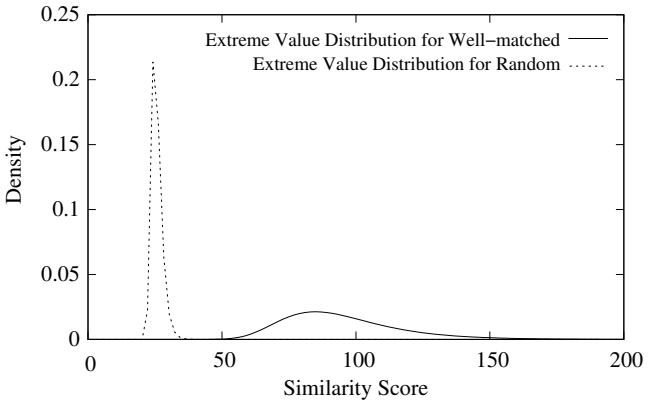

Figure 2. Distributions of Smith-Waterman scores using our transformation (with no mask) and the adding method for determining significance. Curves generated from 10,000 comparisons with base sequence length between 600 and 800, matching portion length 300, and with well-matched sequences suffering an average of 52.5 substitutions and 52.5 indels.

than what could possibly be gleaned from the scores returned by an unmodified Smith-Waterman implementation. This corresponds to asserting that the adversary cannot, using our scheme, learn any more about a private sequence U than what is revealed by the unmodified Smith-Waterman scores of alignments of U with the sequences in $\mathcal{B}$. This is clearly not satisfied by our transformation, since it leaks information: though the public databases are large (GenBank, for example, is estimated to contain more than 54 billion bases in 50 million sequences as of this writing), we must assume that an adversary can completely determine the contents of $\mathcal{B}$, and will thus know the locations of all instances of a single nucleotide literal in U.

Entropy calculations give a rough estimate of the amount of information leaked. Assuming that C_δ is the number of instances of literal δ in the length-N sequence U, then the conditional entropy of U given $F(U, \delta)$ is $(N - C_\delta) \log 3$ (justification in Appendix B). Since the entropy of U is $2N$, approximately $2N - (N - C_\delta) \log 3$ bits of information are leaked. Thus, for example, given a sequence of 600 nucleotides in which $1/4$ are δ, the entropy is 1200 and the conditional entropy is $450 \log 3 \approx 713.23$. That is, approximately $1200 - 713 = 487$ bits of information are leaked by the scheme, with roughly 713 bits of uncertainty remaining.

Implicit in our use of entropy in this context is the assumption that nucleotides in a string are *randomly* distributed according to some predetermined relative frequency. This is not accurate in general. Many genomes have multiple "repeat regions" as well as individual patterns (such as "ATATATAT") that appear several times. It is conceivable that an adversary could exploit this property to gain information about the position of multiple nucleotide literals in parts of the sequence. Such patterns, however, can be (and often are) in practice removed from sequences prior to analysis, because apart from possibly identifying boundaries of specific regions of the genome, the current wisdom is that they reveal little useful information. An analogy in the context of English language text would be inserting multiple copies of the word "the" between words at random places in an unknown document: identifying copies of the pattern "the" would provide little useful information about the contents of the document. For this reason, many public genome databases offer a "scrubbed" version with such patterns removed.

Regardless of whether nucleotides are randomly distributed, entropy calculations in the present context do not provide provable security because of the possibility (which is difficult to quantify in any meaningful way) that a molecular biologist might be able to infer the values of some nucleotides based on the locations of the known nucleotides. For this reason we do not claim that our transformation provides provable security. The biologists [30] with whom we have consulted, however, believe that in practice, given only the positions of a single nucleotide literal, no additional elements can be inferred. Moreover, they believe that there is no biologically useful information (given biologists' current understanding of the structure and function of the genome) that can be gleaned from a nucleotide sequence in which only the positions of a single nucleotide

literal are revealed.

They are, however, quick to point out that given all of the positions of *two* or more nucleotide literals, the private sequence could likely be almost completely reconstructed. For this reason, variants of our scheme that require the creation of multiple tasks (corresponding to distinct nucleotide literals), are vulnerable to collusion. Specifically, under the assumption that the public sequences in a task can be completely known, it is trivial for an adversary to determine whether two tasks represent the same sets of sequences transformed under different nucleotides. The adversary then knows the positions of more than one nucleotide. This is a significant concern, since current volunteer computing platforms lack an effective method for preventing a single *individual* (as opposed to user name) from obtaining several distinct tasks. Fortunately, under some configurations (e.g., global sequence alignment and local alignment of relatively long sequences) multiple tasks are not necessary. Regardless, though the present paper assumes no collusion, collusion resistance is an important consideration that must be addressed before any scheme designed for these platforms can be considered secure.

The conditional entropy of our transformation can be increased by augmenting the basic scheme so that some elements of the offset sequences are masked. That is, we pick, for each task, a sequence over the set $\{0, 1\}$. Each offset sequence in the task is multiplied, element by element, by the task specific mask. Zero entries in the resulting sequence are removed. The increase in entropy resulting from this is based on the proportion ρ of "1" elements in the mask. We have found that for some common application configurations, $\rho = 0.9$ works well.

Note that in the case in which both of sets $\mathcal{A}$ and $\mathcal{B}$ consist of proprietary data, then the adversary cannot ascertain the identity of the nucleotide used to generate a given offset sequence. Moreover, the potential for the type of black-box analysis mentioned in the previous section is removed.

Properties 2 and 3 given in Section 3 state roughly that under the transformation, significant results remain significant (i.e., they are returned as significant results), while the number of false positives (results returned as significant that would not have been significant in the unmodified computation) remains reasonably small. Using standard statistical inference techniques, extreme value density functions can be fit to the simulated score data. Then, by choosing a threshold that corresponds to a lower percentile of the distribution of well-matched scores, the false positive error rate can be controlled. In fact, as seen in Figures 1, 2, 4, 5, and 6, the degree of overlap between the two distributions is often so minuscule that the probability that a randomly generated sequence has a higher score than a well-matched one is rarely more than 10^{-4}.

The benefit is clear: our scheme maintains Properties 2 and 3 by ensuring that scores for appropriate sequences (i.e., well-matched) are clearly separated from inappropriate sequences.

6. Simulation Results

We tested our scheme using several parameter settings, and for each setting generated four scoring distributions: random unmodified sequences, random modified sequences, well-matched untransformed sequences, and well-matched transformed sequences. In this context, "random" refers to a sequence whose nucleotides are generated at random, where a single probability distribution determines, for all positions, the likelihood that a given nucleotide literal occurs; "well-matched" refers to a sequence that is derived from a given random sequence via a finite number of mutations (i.e., random indels or substitutions), and should match the original sequence better than a different, completely random, sequence; "unmodified" refers to using the original Smith-Waterman algorithm; and "modified" refers to using our modified Smith-Waterman scheme.

"Goodness" of a score is relative to an a priori threshold that quantifies statistical significance. Hence the best matched pair in a specific task is not necessarily statistically significant overall. The efficacy of an algorithm in the present context thus depends on the ability to generate statistically significant scores for sequences that descend from a common ancestor.

Simulation results use data generated from samples of size 10,000. More specifically, 10,000 pairs of random sequences were generated and scored (compared) using the Smith-Waterman algorithm. These same sequence pairs were then modified according to our various transformations and scored again. A similar process was used with 10,000 artificially generated well-matched sequence pairs.

We use the scoring function $s(a, b) = 1$ if $a = b$ and $s(a, b) = -1$ if $a \neq b$, and affine gap penalty $g(k) = 2 + 1(k - 1)$, where k represents gap length. These values are used in practice, and are the same as those used by Waterman [37]. The viability of our strategy depends only on the binary nature of the scoring function (any matching pair of literals receives the same score, as does any pair of mismatched literals), and not on the specific values assigned as scores.

Figures 1 and 2 depict score distributions for random and well-matched sequence pairs transformed according to our basic scheme. Parameters for these experiments were sequence length of 600-800, and the relative frequency of each symbol was 0.25. Well-matched sequences were mutated over 15 "generations" using a 0.01 mutation probability per symbol per generation. With these parameters, mutated sequences are expected to differ from the original in approximately 15% of the symbols before our pri-

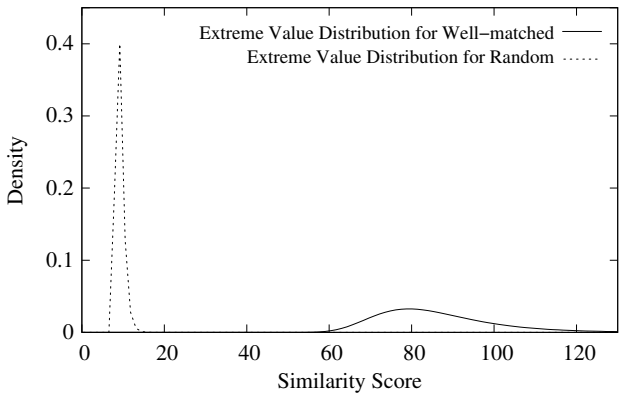

Figure 3. Distributions of Smith-Waterman similarity scores using our transformation (with no mask) and the maximum method for measuring significance and long sequences. Curves generated from 1000 comparisons with base sequence length 2000, matching portion length 1000, and with well-matched sequences suffering an average of 150 substitutions and 150 indels.

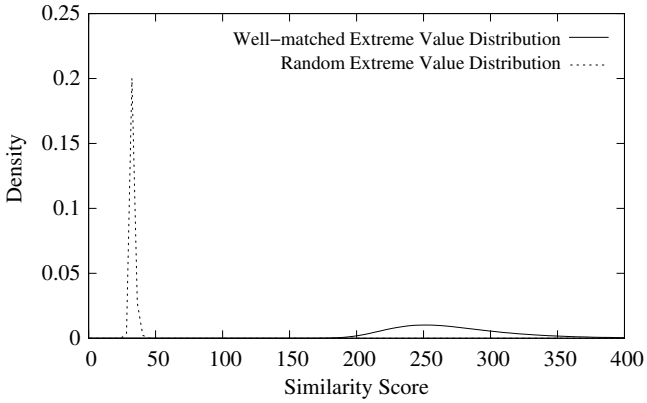

Figure 4. Distributions of Smith-Waterman scores using our transformation (with no mask) and the add method for measuring significance. Curves generated from 1000 comparisons with base sequence length 2000, matching portion length 1000, and with well-matched sequences suffering an average of 150 substitutions and 150 indels.

vacy scheme is applied. Figure 1 represents the "maximum" scoring scheme, in which a single unmodified task is divided into four tasks (each containing the offsets corresponding to a single nucleotide literal), and a result is deemed significant provided the maximum score of the four tasks exhibits a statistically significant match. Figure 2 represents the "adding" scoring scheme, in which the scores from each of the four modified tasks are added to determine significance. The separation between the curves in these figures is a measure of the efficacy of the Smith-Waterman algorithm as applied to transformed sequences. Specifically, the greater the separation between curves, the more like Smith-Waterman our transformed algorithm performs.

These particular experiments represent a worst case: relatively short (from a biological perspective) sequences. Despite this, our statistical analysis indicates that the probability of a transformed well-matched sequence scoring higher than a transformed random sequence is approximately 1.3×10^{-9}. Stated another way, the expected number of trials one would need to run before seeing a random sequence pair score better than a well-matched sequence pair is greater than 7.5×10^{8}.

Scores generated from longer sequences exhibit far greater separation. Figure 3 depicts Smith-Waterman similarity scores using the basic transformation with the maximum method for measuring significance. These curves were generated using length 2000 sequences with matching portions of length 1000. The well-matched sequences experienced an average of 150 substitutions and 150 indels. Figure 4 in Appendix **??** depicts a similar experiment but with the adding method of determining significance. In both of these cases, the probability that a well-matched sequence pair scores less than a random sequence pair is infinitesimally small. The expected number of trials one would need to run before seeing a random pair score better than a well-matched pair is more than 1.8×10^{61}.

Figure 5 depicts score distributions for the basic scheme augmented with masking ($\rho = 0.9$) as applied to sequences of length 1000-1300. Well-matched sequences have matching portion length 500. Here, there is a small probability that a pair of random sequences can score better than a well-matched pair (the overlap of the dashed and solid curves). This can be eliminated entirely if the supervisor is willing to incur some false positives. This is reasonable, since the matches that are missed are at the low end of the spectrum, indicating that a higher proportion of them (if found) would be culled in the postprocessing.

These figures demonstrate that our strategy preserves sufficient information such that similarity between sequences can be accurately measured. More important, they validate the concept of sufficient accuracy: that suitable modification of task procedures can preserve functionality while simultaneously enhancing data privacy.

7. Related work

There are a number of recent studies that focus on the general issue of securing volunteer distributed computations. Golle and Mironov [21] study computations involving inversion of a one-way function (IOWF). These applications seek the pre-image x_0 of a distinguished value y_0 under a one-way function $f : D \rightarrow R$. Golle and Mironov

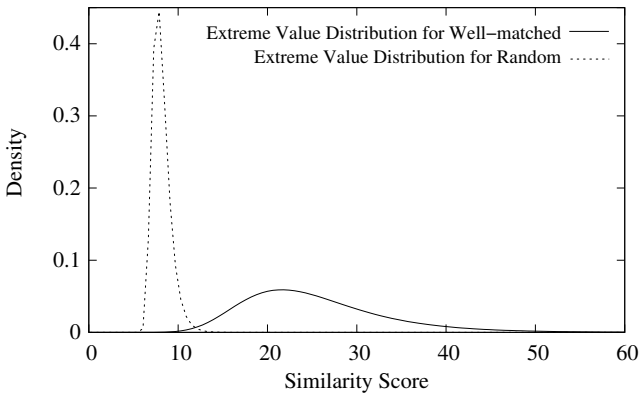

Figure 5. Distributions of Smith-Waterman similarity scores using our transformation and the maximum method for determining significance and augmented with a $\rho = 0.9$ mask. Curves generated from 1000 comparisons with base sequence length between 1000 and 1300, matching portion length 500, and with well-matched sequences suffering an average of 86.25 substitutions and 86.25 indels.

present several variations of a basic *ringer* scheme designed to detect participants who attempt to claim credit for work not completed. Their strategy involves precomputing values of f and planting those results in task data spaces. Since participants are not able to distinguish ringers from true data, the probability of detecting cheating is increased. Szajda, Lawson, and Owen [34] extend the ringers method to handle more general functions, and also present a technique for handling sequential computations, in which a task consists of the repeated application of f to a single input. Golle and Stubblebine [22] and Sarmenta [32] discuss strategies for the intelligent application of redundancy to volunteer distributed computations. The former discusses a security based administrative framework for commercial distributed computations. This provides increased flexibility and protection by varying the distributions that dictate the application of redundancy. Sarmenta instead proposes a credibility-based system in which multiple levels of redundancy are used, with parameters determined by a combination of security needs and participant reputations. Monrose, Wyckoff, and Rubin [25] deal with the problem of guaranteeing that a participant assists in a computation, assuming that the participant's goal is to maximize profit by minimizing cost. Their method involves instrumenting task code at compile-time to produce checkable state points that constitute a proof of execution. Participants return results along with the proof to a verifier, which then runs a portion of the execution and checks it against the returned state checkpoints. None of these papers considers the topic of providing data confidentiality for these computations.

There has been a considerable amount of work in theoretical computer science concerning computing with encrypted data. Feigenbaum [17] examines plausible formal definitions of encryptability, and shows that under one such definition, all NP-complete problems that are polynomially isomorphic to CNF-SAT are encryptable. Abadi, Feigenbaum, and Killian [2, 3] develop a framework for describing in an information-theoretic sense what data information is hidden and what is leaked in a given encryption scheme. Their main encryptability result is that if f can be computed in expected polynomial time with zero error probability, then f is encryptable such that no information about x is leaked. Their protocol, however, requires m rounds of communication, where m is constrained to be polynomial in $|x|$. Interactive confidentiality protocols are impractical in our context because they scale poorly.

Abadi and Feigenbaum [1] describe a two-party protocol for secure circuit evaluation on a general boolean circuit. Secure circuit evaluation protocols provide greater security than what is necessary in our context because they assume that the details of f remain hidden from the owner of the data. Regardless, their method (and others requiring interaction [8, 9, 36]) requires much more server-participant interaction than is practical for a volunteer distributed computation. Sander, Young, and Yung [31] develop a non-interactive protocol called Symmetrically-secure CryptoComputing (SYC), which provides a variant of secure circuit evaluation, hiding the input x and revealing only a bound on the depth of the circuit f. Their solution, however, is limited to log-depth circuits, and is thus impractical in our context.

Rivest, Adleman, and Dertouzos [29] formally introduced the notion of privacy homomorphism, whose existence in theory allows computing with encrypted functions. They conclude that these homomorphisms are inherently limited in their capabilities since comparisons cannot be included in the possible set of operations without creating a vulnerability to ciphertext-only attacks. They also raise the (still open) question of the existence of highly secure privacy homomorphisms that use large sets of operations. Ahituv, Lapid, and Neumann [5] show that if a privacy homomorphism allows the addition operation, then it is insecure under chosen plaintext attacks. Brickell and Yacobi [12] introduce R-additive privacy homomorphisms, which are secure under addition, but place constraints on the number of ciphertexts that can be added. In general, secure privacy homomorphisms that preserve more than one operation are difficult to find. An exception is a homomorphism developed by Ferrer [15] that preserves addition and multiplication while resisting known plaintext attacks. Though elegant, these homomorphisms have far too limited operation sets to be of practical use in volunteer distributed computations.

The problem of multiparty function computation involves players P_i, $1 \leq i \leq n$, with private inputs x_i who wish to evaluate a function $f(x_1, \ldots, x_n)$ without revealing any more information about the x_i than is implicitly contained in the output value. Yao [38] introduces this problem and develops a protocol for the two player case. Goldreich, Micali, and Wigderson [20] extend this result to several parties, developing a protocol that leaks no input information provided a majority of honest players. Both of these results are based on assumptions of intractability of certain functions (the cryptographic approach). Results based on the information-theoretic approach, which does not assume limits on processor computation power, include work of Ben-Or, Goldwasser, and Wigderson [10], who present a protocol that achieves a tight bound on the size of the group of colluding players that can disrupt the computation, and Chaum, Crépeau, and Damgard [14], who show that any "reasonable" multiparty protocol can be achieved if at least 2/3 of the players are honest. Goldreich [19] provides a survey of results in this area. As with the secure circuit evaluation work above, these protocols have communication and computation complexity that precludes their use in the current context.

8. Conclusions

Via a specific application, we have introduced a strategy for enhancing data privacy in some distributed volunteer computations. The strategy is based on the observation that the requirements for computing with obscured data can be much less restrictive in some of these computations than in traditional execution models because of the filtering nature of certain volunteer computations. In particular, because the identification of important data, rather than the output associated with this data, is the goal of these computations, there can be considerable flexibility in task procedure definitions. This flexibility can be leveraged to provide data privacy by allowing transformations to data and procedures that retain sufficient information for filtering, while simultaneously obscuring data details so that identification is difficult, if not impossible.

We illustrated the potential of this strategy by describing a scheme for enhancing data privacy in the Smith-Waterman local sequence comparison algorithm. Our modifications are a promising first step, in that they provide reasonable, though not rigorously provable, data privacy while preserving sufficient information for distinguishing well-matching sequences. In addition, by presenting a practical, important, and non-trivial real-world application that requires privacy and is efficiently parallelizable, we have begun to populate a potential benchmark suite of applications for privacy study.

Acknowledgments

We would like to thank biologists Rafael De Sa, Laura Runyen-Janecky, and Joe Gindhart, all from the University of Richmond, for their patient and thorough treatment of our questions. We would also like to thank the anonymous reviewers and Tadayoshi Kohno for many valuable comments that helped make this a better paper than it otherwise would have been.

References

[1] M. Abadi and J. Feigenbaum. Secure circuit evaluation: A protocol based on hiding information from an oracle. *Journal of Cryptology*, 2(1):1–12, 1990.

[2] M. Abadi, J. Feigenbaum, and J. Kilian. On hiding information from an oracle. In *Proceedings of the 19th ACM Symposium on the Theory of Computing*, pages 195–203, 1987.

[3] M. Abadi, J. Feigenbaum, and J. Kilian. On hiding information from an oracle. *Journal of Computer and System Sciences*, 39(1):21–50, August 1989.

[4] M. Abdalla, M. Bellare, D. Catalano, E. Kiltz, T. Kohno, T. Lange, J. Malone-Lee, G. Neven, P. Paillier, and H. Shi. Searchable encryption revisited: Consistency properties, relation to anonymous ibe, and extensions. Cryptology ePrint Archive, Report 2005/254, 2005. `http://eprint.iacr.org/`.

[5] N. Ahituv, Y. Lapid, and S. Neumann. Processing encrypted data. *Commun. ACM*, 30(9):777–780, 1987.

[6] M. Ajtai and C. Dwork. A public-key cryptosystem with worst-case/average-case equivalence. In *Proceedings of the 29th Annual ACM Symposium on Theory of Computing*, pages 284–293, El Paso, Texas, May 1997.

[7] S. Altschul and W. Gish. Local alignment statistics. *Methods Enzymol*, 266:460–480, 1996.

[8] J. Bar-Ilan and D. Beaver. Non-cryptographic fault-tolerant computing in a constant number of rounds of interaction. In *Proceedings of 8th ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing*, pages 201–209, 1989.

[9] D. Beaver, S. Micali, and P. Rogaway. The round complexity of secure protocols. In *Proceedings of the Twenty-second Annual ACM Symposium on Theory of Computing*, pages 503–513. ACM Press, 1990.

[10] M. Ben-Or and A. Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation. In *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*, pages 1–10, 1988.

[11] M. Bishop. *Computer Security: Art and Science*. Addison-Wesley, 2003.

[12] E. Brickell and Y. Yacobi. On privacy homomorphisms (extended abstract). In D. Chaum and W. Price, editors, *Advances in Cryptology—EUROCRYPT '87*, volume 304 of *Lecture Notes in Computer Science*, pages 117–126, Berlin, 1987. Springer-Verlag.

[13] E. Chargaff. Structure and function of nucleic acids as cell constituents. *Fed. Proc*, 10:654–659, 1951.

[14] D. Chaum, C. Crépeau, and I. Damgard. Multiparty unconditionally secure protocols. In *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*, pages 11–19, 1988.

[15] J. Domingo-Ferrer. A new privacy homomorphism and applications. *Information Processing Letters*, 60(5):277–282, December 1996.

[16] C. Dwork, M. Naor, and O. Reingold. Immunizing encryption schemes from decryption errors. In C. Cachin and J. Camenisch, editors, *Advances in Cryptology – EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 342–360, Interlaken, Switzerland, May 2004. Springer-Verlag.

[17] J. Feigenbaum. Encrypted problem instances, or... Can you take advantage of someone without having to trust him? In *Proceedings of Crypto' 85*, pages 477–488. Springer-Verlag, 1986.

[18] Genbank. `http://www.ncbi.nlm.nih.gov/GenBank/GenBankOverview.html`.

[19] O. Goldreich. Secure multi-party computation. Working Draft, 2000.

[20] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game. In *Proceedings of the Nineteenth Annual ACM Conference on Theory of Computing*, pages 218–229, 1987.

[21] P. Golle and I. Mironov. Uncheatable distributed computations. In *Proceedings of the RSA Conference 2001, Cryptographers' Track*, pages 425–441, San Francisco, CA, 2001. Springer.

[22] P. Golle and S. Stubblebine. Secure distributed computing in a commercial environment. In P. Syverson, editor, *Proc. of Financial Crypto 2001*, volume 2339 of *Lecture Notes in Computer Science*, pages 289–304. Springer-Verlag, 2001.

[23] S. Karlin and S. Altschul. Methods for assessing the statistical significance of molecular sequence features by using general scoring schemes. *Proceedings of the National Academy of Sciences (USA)*, 87:2264–2268, March 1990.

[24] S. Karlin and S. Altschul. Applications and statistics for multiple high-scoring segments in molecular sequences. *Proceedings of the National Academy of Sciences (USA)*, 90:5873–5877, June 1993.

[25] F. Monrose, P. Wyckoff, and A. Rubin. Distributed execution with remote audit. In *Proceedings of the 1999 ISOC Network and Distributed System Security Symposium*, pages 103–113, 1999.

[26] S. Needleman and C. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology*, 48:443–453, 1970.

[27] W. Pearson. Empirical statistical estimates for sequence similariy searches. *Journal of Molecular Biology*, 276:71–84, 1998.

[28] T. Phan, L. Huang, and C. Dulan. Challenge: Integrating mobile wireless devices into the computational grid. In *Proceedings of the Eight International Conference on Mobile Computing and Networking (MobiCom 2002)*, pages 271–278, Atlanta, GA, September 2002.

[29] R. Rivest, L. Adleman, and M. Dertouzos. On data banks and privacy homomorphisms. In R. D. Millo, D. Dobkin, A. Jones, and R. Lipton, editors, *Foundations of Secure Computation*, pages 169–179. Academic Press, New York, 1978.

[30] R. D. Sa, L. Runyen-Janecky, and J. Gindhart, July 2005. Personal conversations.

[31] T. Sander, A. Young, and M. Yung. Non-interactive cryptocomputing for NC^1. In *IEEE Symposium on Foundations of Computer Science*, pages 554–567, 1999.

[32] L. Sarmenta. Sabotage-tolerance mechanisms for volunteer computing systems. *Future Generation Computer Systems*, 18(4):561–572, March 2002.

[33] T. Smith and M. Waterman. Identification of common molecular subsequences. *Journal of Molecular Biology*, 147:195–197, 1981.

[34] D. Szajda, B. Lawson, and J. Owen. Hardening functions for large-scale distributed computations. In *Proceedings of the 2003 IEEE Symposium on Security and Privacy*, pages 216–224, Berkeley, CA, May 2003.

[35] D. Szajda, B. Lawson, and J. Owen. Toward an optimal redundancy strategy for distributed computations. In *Proceedings of the 2005 IEEE International Conference on Cluster Computing (Cluster 2005)*, Boston, MA, September 2005.

[36] S. Tate and K. Xu. On garbled circuits and constant round secure function evaluation. Technical Report TR 2003-02, University of North Texas, Computer Privacy and Security (CoPS) Lab, 2003.

[37] M. Waterman. *Introduction to Computational Biology: Maps, Sequences, and Genomes*. Interdisciplinary Statistics. Chapman & Hall, 1995.

[38] A. Yao. How to generate and exchange secrets. In *Proceedings of the 27th Annual Symposium on Foundations of Computer Science*, pages 162–167, Toronto, Canada, October 1986.

A. Sequence Comparison Algorithm Details

The details of the Smith-Waterman global sequence alignment algorithm follow. We assume here, for simplicity of notation, that length n sequences begin at index 1 and end at index n. Recall that S denotes the similarity score of a sequence pair, s denotes the similarity function for symbols, and g denotes the gap penalty.

Theorem. [37]. If $U = u_1 u_2 \dots u_n$ and $V = v_1 v_2 \dots v_m$, define

$$S_{i,j} = S(u_1 u_2 \dots u_i, v_1 v_2 \dots v_j).$$

Also, set

$$S_{0,0} = 0, \quad S_{0,j} = \sum_{k=1}^{j} g(v_k), \text{ and } S_{i,0} = \sum_{k=1}^{i} g(u_k).$$

Then

$$S_{i,j} = \max\{S_{i-1,j} + g(u_i), S_{i-1,j-1} + s(u_i, v_j), S_{i,j-1} + g(v_j)\}.$$

Proving the validity of the dynamic programming approach in this context is straightforward. One need only observe that an alignment ending with indices i and j must end with one of the choices below.

$$\begin{array}{lll} \cdots u_i & \cdots u_i & \cdots - \\ \cdots - & \cdots v_j & \cdots v_j \end{array}$$

Thus the best alignment ending with indices i and j must be the best alignment ending with indices i and $j-1$ plus the gap penalty, or the best alignment ending with indices $i-1$ and $j-1$ plus $s(u_i, v_j)$, or the best alignment ending with indices $i-1$ and j plus the gap penalty.

For local sequence comparison, define for (i, j) pair the function H by

$$H_{i,j} = \max\{0; \\ S(u_x u_{x+1} \ldots u_{i-1} u_i, v_y v_{y+1} \ldots v_{j-1} v_j) : \\ 1 \le x \le i,\ 1 \le y \le j\}.$$

Then H can be computed using the following two results from [37].

Theorem. Assume that the gap function g is a function of gap length. Set $H_{0,0} = 0$, and set $H_{i,0} = H_{0,j} = 0$ for $1 \le i \le n$ and $1 \le j \le m$. Then

$$H_{i,j} = \max\Big[0, \max_{1\le k\le i}\{H_{i-k,j} - g(k)\}, \\ H_{i-1,j-1} + s(u_i, v_j), \max_{1\le l\le j}\{H_{i,j-l} - g(l)\}\Big].$$

The proof is similar to that of the global alignment algorithm.

Finally, we have the following

Corollary.

$$H(U, V) = \max\{H_{k,l} : 1 \le k \le n,\ 1 \le l \le m\}.$$

B. Entropy Calculation

Given that the adversary has determined the location of all instances of a single nucleotide, we can measure conditional entropy. Assume that our original sequence U has length N, and that the adversary has been provided with $F(U, \delta)$ for some fixed literal $\delta \in \Sigma$. There are 4^N possible length N sequences over Σ, and we may assume that these are enumerated such that each has a unique integer index in the range 1 to 4^N inclusive. That is, all possible length N sequences over Σ occur exactly once among the set $\mathcal{S} = \{S_1, S_2, \ldots, S_{4^N}\}$. Let X be the random variable that has a uniform distribution over $\mathcal{S}$. (Technically, X is the random variable that has a uniform distribution over the set of integers between 1 and 4^N inclusive.) The entropy [11], $H(X)$, of X is easily shown to be $2N$, which is to be expected, since in effect each nucleotide contains two bits of uncertainty. Now let us consider the conditional entropy $H(X|Y)$ of X given that the adversary has received offset sequence $Y = F(U, \delta)$. Let C_δ denote the number of occurrences of literal δ in U. Then

$$H(X|Y) = \\ -\sum_{i=1}^{4^N} P(X = S_i | Y = F(U,\delta)) \quad \times \\ \log(P(X = S_i | Y = F(U,\delta))).$$

Since, however, the positions of literal δ are revealed by $F(U, \delta)$, and δ is known to the adversary, thus there are 3^{N-C_δ} sequences in $\mathcal{S}$ that could be the preimage of U. Let $k_1, k_2, \ldots, k_{3^{N-C_\delta}}$ be the indices (over the set $\mathcal{S}$ of these possible preimages). Thus,

$$H(X|Y) = \\ -\sum_{i=1}^{3^{N-C_A}} P(X = S_{k_i} | Y = F(U,\delta)) \quad \times \\ \log(P(X = S_{k_i} | Y = F(U,\delta))) \\ = -3^{N-C_A} \frac{1}{3^{N-C_A}} \log\left(\frac{1}{3^{N-C_A}}\right) \\ = (N - C_A)\log 3.$$