DNA-INSPIRED INFORMATION CONCEALING

LUKAS KENCL AND MARTIN LOEBL

Abstract. Protection of the sensitive content is crucial for extensive information sharing. We present a technique of information concealing, based on introduction and maintenance of families of repeats. Repeats in DNA constitute a basic obstacle for its reconstruction by hybridisation. Information concealing in DNA by repeats is considered in [1].

1. Introduction

Contemporary computer systems may be distributed and may consist of many interconnected processing units or a large number of networked computer subsystems. In addition contemporary digital networks may consist of a large number of end- and intermediate- nodes. In all these systems, information, in the form of the sequences over some alphabet of symbols, is circulating or being stored. The entity controlling a subsystem or a node is often unwilling or prohibited to share this information-sequences with other nodes. However, sharing of some reduced local information might be very useful for purposes of security, stability and various analysis of the system performance, and for data mining. Such analysis might for example allow to identify frequently appearing segments by performing approximate statistical analysis on segment frequency, allowing to detect replicating malicious code-worms. It also allows to identify segments-markers of computer viral infection, by detecting patterns existing in some database of malicious sequences. Such databases are used e.g. in contemporary intrusion detection systems or spam filters. It has been shown that being able to perform pattern matching against only fixed-length prefixes or substrings of longer sequences can provide approximate hints as to the presence of suspicious content [2]. Likewise, established worm detection techniques such as Autograph [3] or EarlyBird [4] are based on counting frequency of small blocks of a fixed size.

Sharing of reduced local information among the members of an interconnected computer system or communication network thus helps to discover attacks earlier. Affected parts may be isolated and further

M. Loebl is with the Department of Applied Mathematics and Institute of Theoretical Informatics (ITI), Charles University, Prague, Czech Republic.

L. Kencl is with the Research and Development Centre (RDC), Czech Technical University, Prague, Czech Republic.

attack spread prevented. The benefits of sharing local information may be reaped in case of existence of a computational information processing, which preserves local information (e.g. all segments of certain maximal length) and makes impossible to reconstruct longer or sensitive parts of the information sequences.

We call such information processing *concealing*. The systems which conceal information and share the concealed information are likely to possess a competative advantage in the form of robustness, attack resistance and immunity due to ability to exchange, publish and protect information. Clearly, any information concealing algorithm needs to address two conflicting goals:

(1) preserving *presence* and, possibly, *frequency rank* of segments of given size (making spam identification and worm detection still possible), while
(2) making reconstruction of content longer than the predefined limit computationally hard (e.g. disabling interpretation or understanding of the private content).

1.1. Main contribution.

The main contribution of this paper is

- Formulation of the information concealing problem
- Presentation of an information concealing algorithm
- Analysis of the algorithm and a proof of the hardness of reconstruction of the input sequence

2. Related Work

2.1. Repeats in DNA.

Our inspiration comes from an important feature of eukaryotic DNA, namely that it contains various *repeat families*, and that their presence constitutes a basic difficulty in DNA reconstruction by hybridisation [6].

A large proportion of eukaryotic genomes is composed of DNA segments that are repeated either precisely or in variant form more than once. Highly repeated segments are arranged in two ways: as tandem arrays or dispersed among many unlinked genomic locations. As yet, no function has been associated with many of the repeats [8]. In the paper [1] which accompanies this paper, the authors propose that in eukaryotes the cells have DNA as a depositary of concealed genetic information and the genome achieves the self-concealing by accumulation and maintenance of repeats. The protected information may be shared and this is useful for the development of intercellular communication and in the development of multicellular organisms.

The assertion that the repeats are maintained in DNA in a programmed way for self-concealing explains basic puzzling features of repeats: the uniformity along with the polymorphism of the repeated sequences; the freedom of the repeated DNA to adopt quite different primary sequences in closely related species; apparent non-functionality of the precise amount or the precise sequence of the repeats.

The containment of repeats versus DNA sequencing problem is receiving extensive attention of biologists, computer scientists and mathematicians (see [5], [6], [7]).

2.2. **Repeats versus DNA reconstruction.** We explain the basic idea of concealing by repeats in this subsection. Assume we are given a collection $\mathcal{K}$ of segments of DNA. Each segment S from $\mathcal{K}$ is divided into two parts, the initial part $S(I)$ and the terminal part $S(T)$. We thus may write $S = S(I)|S(T)$. This is an artificial assumption imposed only for the clarity of the presentation.

A *reconstruction* of $\mathcal{K}$ is a sequence of its segments so that the terminal part of each segment agrees with the initial part of next segment in the sequence. If several of these initial and terminal parts coincide, there may be an exponential number of possible reconstructions.

Let us consider a very simple example. Let $\mathcal{K}$ be the following collection of segments, where the initial and the terminal parts are divided by the vertical line:

$$A|B, B|A, A|C, C|A, B|C, C|B.$$

the following sequences are some of the possible reconstructions:

$$ABACBCA, ACABCBA, BACABCB, ABCACBA.$$

In this simple example, even unlimited computational power is useless to anybody who wants to obtain the correct reconstruction from the many possible reconstructions. This phenomenon may well be described in terms of the de Bruin graph: this graph has a node for each segment which is an initial or a terminal part of an element of $\mathcal{K}$. For each segment S of $\mathcal{K}$ there is an arrow (a directed edge) from $S(I)$ to $S(T)$.

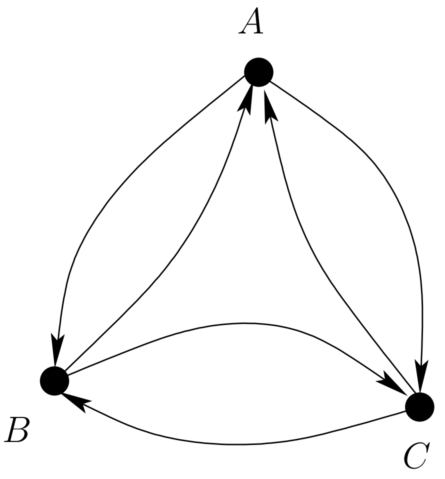

Figure 1. De Bruin graph for $\mathcal{K}$

The possible reconstructions now correspond to the walks on the de Bruin graph so that each directed edge is traversed exactly once. These walks are usually called Euler walks. If a node of the de Bruin graph has more than one outgoing incident directed edge, then locally there are several independent ways to traverse these edges. The number of the Euler walks of the de Bruin graph is therefore typically exponential in the number of these nodes (see [7] for the calculations).

2.3. Concealing in Information and Communication Technologies.

The concept of hiding private or sensitive data but preserving some form of structural information has been studied recently in various sub-domains of ICT. Some techniques concentrate on hiding the originator of information, i.e. *anonynimization*, other focus on enabling particular functions over the data that can be shared among multiple partners, such as *private matching.*

2.3.1. *Concealing Network Data.*

An anonymization scheme over the network packet *IP addresses* called CryptoPan [11] preserves the prefix hierarchy of the original addresses, while making them computationally hard to reconstruct by using hashing. This in turn allows to share network traces (with packet headers only), with preservation of the prefix hierarchy.

Similarly, in [12], structure of the router configuration files and data is preserved, while the actual values are obfuscated.

A technique to process and transform the network *packet payload* has been proposed in [13]. This method uses dictionaries of important sequences that are valuable from the data mining perspective and should be preserved, while encrypting the rest of the information with cryptographically strong hash function. This technique performs well in terms of data protection, however, it only allows to study content portions pre-determined by a known list, and thus does not allow to study the payload to detect previously unknown content, such as e.g. malicious subsequence.

The popular Bloom filter [14] approach is used in constructing the Hierarchical Bloom Filter *payload attribution* technique [15]. A Bloom filter can store (incompletely but efficiently) input items (which can be substrings) and easily answer set membership queries. It consists of k hash functions, each associating one of m numbers to each input item. Set membership queries exhibit no false negatives, but can have false positives.

Payload attribution with a Hierarchical Bloom Filter stores segments of network packet payloads with their IP source and destination addresses. Each payload is cut into segments $s_1, \dots, s_n$. The s_i's are stored in a Bloom filter of level 0, *the pairs* $s_1 s_2, s_3 s_4, \dots$ in Bloom filter of level 1, quadruples in level 2, and so on. A query on an excerpt of payload, which may consist of several consecutive blocks, may answer the source and destination address by running through consecutive hierarchy levels.

The authors propose deployment at network concentration points. Privacy protection is to be achieved by restricting access of entities that can pose queries, otherwise exhaustive attacks might lead to payload reconstruction.

2.3.2. *Private matching.* Private matching [16, 17] focuses on the problem of two entities trying to find common data elements in their databases, without revealing private information. The basic property (and difference to the general information concealing problem) is that only two parties are involved; a multiparty solution is a future work suggestion. Further problems are asymmetry in the sequence of information exchange among the parties and needed presumption of honesty ('semi-honesty' in the paper).

Private matching is a special case of cryptography theory of *multi-party computation*: m parties want to compute function f on their m inputs. In the *ideal model*, where a trusted party exists, the parties give their inputs to the trusted authority, it calculates f, and returns the result to each party. The ideal model assumes an ideal situation: for example, no protocol can prevent a party to change its input before the communication is started. A secure multiparty computation protocol *emulates* what happens in the ideal model.

Paper [16] also introduces 'data ownership certificates' to modify the private matching protocols to be unspoofable. This technique is shown to be useful in a more practical setting to enable privacy-protecting sharing of e-mail white-lists in [18].

2.3.3. *Data Masking.* Various techniques of masking, sanitizing and obfuscating data have been studied to enable test- or third-party development over sensitive databases (such as the Human Resources data). After sanitization, the database remains usable - the look-and-feel and some relations and distributions are preserved - but the information content is secure. The used techniques include masking, shuffling, substitution, number-variance, encryption etc [19]. These techniques share a similar goal with information concealing, but focus on structured data without the need of preserving the local information.

2.3.4. *Data Mining and Anonymization.* In data mining, anonymization mechanisms (obfuscating the originator or the private part of the data) are currently studied intensively. Privacy mechanisms can be classified into several categories, according to where they are deployed during the life cycle of the data. The mechanism proposed in this paper falls into the category where the individuals trust no one but themselves, and they conceal their respective data before they make them available for sharing. The existing algorithms in this category [20, 21, 22, 23, 24] are called local perturbation; they are based on different ideas then the concealing procedure proposed in this letter.

In another category, data publishing, data are anonymized at a central server; the individuals are required to trust this server [25]. Anonymization in social networks is studied in [26].

An important theoretical foundation for data anonymity and originator protection was laid in [29]. The k-anonymity model for protecting privacy allows holders to release their private data without being distinguishable from at least k-1 other individuals also in the release.

2.3.5. *Steganography.* This form of information hiding [27, 28] is a related art and science of writing hidden messages in such a way that no one apart from the sender and intended recipient realizes there is a hidden message; this nowadays includes concealment of digital information within computer files. Comparably, steganalysis is the art of detecting the hidden information.

Steganography is a mature science, in particular focussing on the domain of Digital Rights Management (DRM), where various 'watermarking' or 'tamper-proofing' techniques may seamlessly embed extra information about the origin of a digital work within itself. This is a different goal than the proposed information concealing. While the embedded information may be well concealed and thus very hard to reconstruct, data mining of such information would not be generally possible. However, it would be very interesting to apply the steganalysis tools to the information concealing 'attacker problem' (see Section 3) and it certainly belongs among our future work.

2.3.6. *Information Retrieval.* The "attacker problem" of concealed string reconstruction (see Section 3) has a strong connection to the problem of information retrieval [30], where probabilistic information about the expected string (e.g. natural text) may be used to derive further information or assist text reconstruction.

2.4. **Segments shuffling.** Finally we mention that our first attempt to solve the anonymization problem [31] was using random permutations of a collection of short overlapping segments. This method however by itself doesnot lead to concealing the original data information. It is shown in this paper that in order to sufficiently extend the families of repeats of the resulting sequence and make the concealing successful, other procedures need to be performed as well. In particular the overlapping segments containing complete local information need to be prolonged by attaching additional short segments to their beginning and/or their end. The shuffling permutation also needs to satisfy some properties. This is described in the rest of the paper.

3. Information Concealing Problem

We introduce formally the information-sequence concealing problem. Let $|\omega|$ denote the number of symbols (length) of sequence ω. The *sequence concealing* problem is the following: Given a sequence ω and a small positive integer k, we want to transform ω to another sequence ω_F so that:

I.- If s is a segment of ω with $|s| \leq k$, then s is a segment of ω_F.
II.- It is computationally hard to reconstruct sequence ω from ω_F.
III.- The length of ω_F is linear in $|\omega|$.
IV.- It is also desirable that with low probability, a segment not in ω appears in ω_F, and that relative frequency (i.e., frequency rank) of segments of ω of a given length is preserved in ω_F. The precise statement of these two conditions is however strongly application dependent.

Given the statement of the information concealing problem, the key issue is how much information about ω can an attacker deduce from ω_F; let us call this issue the *attacker problem.*

Clearly, the answer to the attacker problem is application-dependent. If the input sequence ω is very restrictive, e.g. if a short prefix uniquely determines larger part of ω and the k-segments of ω may be distinguished within the larger k-segment superset of ω_F, then inevitably large part of input ω may be reconstructed from ω_F. In quite a number of practical situations (DNA sequence, computer program, sound and video trace, text on non-specific topic), however, this is not the case. Moreover, for restrictive input sequences, we can perform preparatory procedures (as procedure S described below) which make the input sequence less specific.

This partly justifies the following *consistency assumption* concerning the attacker problem which we need to make in order to carry the security analysis of the concealing algorithm.

Proposition 3.1. *The complete input of the attacker problem, i.e. all the useful information an attacker has about ω, is ω_F, the length $|\omega|$, the length k of the preserved segments and the concealing algorithm used in obtaining ω_F.*

Thus, an attacker may use the list of frequencies of repeats of segments of ω_F along with the knowledge of the concealing algorithm to attempt the reconstruction of ω.

4. Concealing by repeats

The input of the problem is a sequence over an alphabet. We first turn it into a cyclic sequence by connecting its beginning and end.

Next we describe five procedures which are used in the algorithm. The basic pattern of all the procedures is the same and may be described as follows: the input is a cyclic sequence ω. First, ω is partitioned into consecutive disjoint *blocks*. Then the terminal part of the preceding block of length o (the *overlap*) is added in front of each block. The resulting segments contain all the studied local information; depending on the procedure, these segments will also contain some excess information which is vital in a proposed composition of the procedures which forms our concealing algorithm. Next, a segment called *dust* can but neednot be added behind each segment. The enhanced blocks are called the *cards*. The last step consists in arranging the cards into the output cyclic sequence ω_F.

The first procedure S has a preparatory character in the concealing algorithm. Several runs of S have the role of breaking the local sequential order in the input sequence.

4.1. **Procedure** $S(\omega, o, lb, ub)$**.** Its input is a cyclic sequence ω, and it has parameters o, lb, ub; o stands for the size of the overlap, lb is for lower bound of the length of a block, and ub is for the upper bound of the length of a block. The procedure $S(\omega, o, lb, ub)$ is defined by 1.-4. below.

1. We partition (sometimes we say that we *cut*) ω into consecutive disjoint *blocks* $P_1, \ldots, P_m$ such that the length of each P_i is chosen at random between lb, ub.
2. We add overlap of length o in front of each block. The overlapping segments thus contain all the original sub-segments of length up to $o + 1$.
3. The blocks enhanced by the overlaps now start and end with the corresponding overlaps. If these were arranged into a cyclic sequence, the overlaps would neighbor. This may help an attacker in reconstruction. To break the neighborhood relationship of the overlaps, we may add dust (a randomly chosen segment) behind each block. Adding dust is optional and application dependent. A natural restriction is that the dust is a segment of the input sequence and that the average length of dust is $1/2(lb + ub) - o$ to match the average length of the segments complementing the overlaps. However, depending on applications, and the stringency of condition $[IV]$ of the sequence concealing problem, length of dust may be different and the dust need not be a segment of the input sequence.
4. We arrange the resulting cards randomly into a cyclic sequence.

As an illustration we perform S on an example input sequence:

Example 1: Procedure $S(\omega, o, lb, ub)$

Input $\omega =$ ’**the aim of this paper is to present an information concealing algorithm**’, parameters $o = 3$, $lb = 4$, $ub = 6$.

1. First, the input sequence is partitioned randomly into blocks of length $4, 5$ or 6. The blocks are divided by ’+’ below:
   ’**the ai+m of t+his +pape+r is +to pr+esent+ an i+nfor+matio+n co+ncea+ling +algor+ ithm+**’
2. Next we add overlap (of length $o = k - 1 = 3$) in front of each block:
   ’**thmthe ai+ aim of t+f this +is pape+aper is +is to pr+ present+ent an i+n infor+formati+tion co+ concea+cealing +ingalgor+gorithm+**’
3. Next we add the dust behind each block (of length approximately 2), and we get the cards:
   ’**thmthe aip+ aim of tim+f this con+is pape in+aper is a+is to pro p+ presentese+ent an ilgo+n infori +formatifo+tion co + concea ci+cealing pa+ingalgor p+gorithmap+**’
4. Finally the output is given by arranging the cards in a random order (here we use the order $14, 9, 10, 13, 5, 3, 12, 1, 6, 4, 7, 11, 8, 15, 2$):
   ’**ingalgor pn infori formatifocealing paaper is af this conconcea cithmthe aipis to pro pis pape in presentesetion co ent an ilgogorithmap aim of tim**’

4.2. **Procedure** $S^1(\omega, lb, ub)$**.** Procedure $S^1(\omega, lb, ub)$ is as S but the overlap is always the whole preceding block - typically exceeding the size needed to preserve the studied local information (this excess is used in the composition of the procedures forming our concealing algorithm). Hence, if the blocks are

$$\omega = P_1 P_2 P_3 \dots P_m,$$

then the cards of S^1 are $P_1P_2, P_2P_3, \dots, P_mP_1$.

Each P_i appears once as initial segment and once as terminal segment of each card. Hence, the cyclic consecutive order of the cards of S^1 may be described by a permutation π of $1, \dots, m$; for further discussions it turns out useful to define such permutation so that it assigns, to each terminal block of a card, the initial block of the next card. By *permutation of* $1, \dots, m$ we mean a bijection from set $\{1, \dots, m\}$ onto itself. If π is a permutation then π^{-1} denotes the inverse permutation ($\pi(x) = y$ if and only if $\pi^{-1}(y) = x$). Hence, in our formalism, card $P_{i-1}P_i$ is followed by card $P_{\pi(i)}P_{\pi(i)+1}$.

The output of S^1 thus always has form

$$P_1 P_2 P_{\pi(2)} P_{\pi(2)+1} \dots P_{\pi^{-1}(1)-1} P_{\pi^{-1}(1)}.$$

For instance, if we have $m = 3$ then the cards are P_1P_2, P_2P_3, P_3P_1 and a shuffling which results in sequence $P_1P_2P_3P_1P_2P_3$ is described by permutation $\pi(1) = 2, \pi(2) = 3, \pi(3) = 1$.

4.2.1. *Acceptable permutations.* For our purposes, not all permutations π are acceptable; let us formally denote by $\mathcal{A}$ the set of all the *acceptable permutations.* To define $\mathcal{A}$, we first introduce an auxiliary bipartite graph $G(\pi)$.

Definition 4.1. Graph $G(\pi)$ has vertex-set $V = V_1 \cup V_2$ where $V_1 = \{u_1, \ldots, u_m\}$ and $V_2 = \{v_1, \ldots, v_m\}$. The edge-set of $G(\pi)$ is the union of three disjoint perfect matchings of the vertex-set, namely:

1. The perfect matching M_1 consisting of the edges $\{u_i, v_i\}$.
2. The perfect matching M_2 consisting of the edges $\{u_{i+1}, v_i\}$.
3. The perfect matching M_3 consisting of the edges $\{u_{\pi(i)}, v_i\}$.

Definition 4.2. We construct a directed graph $G'(\pi)$ from $G(\pi)$ by first directing each edge of $M_2 \cup M_3$ from V_2 to V_1, and then contracting each edge of M_1.

Definition 4.3. (of set $\mathcal{A}$ of all acceptable permutations) Permutation π is acceptable ($\pi \in \mathcal{A}$) if and only if the following two conditions are satisfied:

1. The directed graph $G'(\pi)$ has a directed eulerian closed walk where the edges of M_2 and M_3 alternate. This condition is equivalent to saying that permutation π describes a rearrangement of the cards of S^1 into a sequence.
2. In the auxiliary graph $G(\pi)$, the union of the perfect matchings $M_2 \cup M_3$ contains many (at least m/c where $c \geq 2$ is a small constant) cycles. This condition is added in order to make the reconstruction of the input sequence hard; see the sections below.

The following observation about the graph $G(\pi)$ will be used in the analysis of the attacker problem.

Observation 4.4. *Let $G(\pi)$ be as in Definition 4.1. For $v_i \in V_2$ let $s(v) = P_i$ be its associated segment. Then we have the following equality between cyclic sequences:*

$$P_1P_2P_3 \ldots P_m = s(M_2(1))s(M_2(2)) \ldots s(M_2(m)),$$

where $M_2(i)$ denotes the vertex of V_2 connected with $u_i \in V_1$ by an edge of M_2.

For illustration we perform S^1 on the output sequence of the previous example (which would be the natural use of S^1, as described later):

Example 2: Procedure $S^1(\omega, lb, ub)$

Input $\omega =$ ’**ingalgor pn infori formatifocealing paaper is af this conconcea cithmthe aipis to pro pis pape in presentesetion co ent an ilgogorithmap aim of tim**’, parameters $lb = 6$ and $ub = 8$.

First, the input sequence is partitioned randomly into blocks of length $6, 7$ or 8. The blocks are divided by ’+’ below:

’**ingalgo+r pn inf+ori for+matifo+cealing+ paape+r is a+f this c+onconce+a cithm+the aipi+s to pro+ pis p+ape in+ present+esetion +co ent +an ilg+ogorith+map ai+m of tim+**’

Next we add overlap in front of each block. For procedure S^1 the overlap is always the whole preceding block. We get the following cards; to make the example easier to understand we indicate by ’*’ the division of each card into two blocks:

’**m of tim*ingalgo+ingalgo*r pn inf+r pn inf*ori for+ori for*matifo+matifo*cealing+cealing* paape+ paape*r is a+r is a*f this c+f this c*onconce+onconce*a cithm+a cithm*the aipi+the aipi*s to pro+s to pro* pis p+ pis p*ape in+ape in* present+ present*esetion +esetion *co ent +co ent *an ilg+an ilg*ogorith+ogorith*map ai+map ai*m of tim+**’

Finally the output is given by rearranging the cards by an acceptable permutation, i.e. by a permutation whose corresponding bipartite graph consists of a lot of cycles. The smallest length of a cycle is 4. It is not difficult to see that the following permutation π creates nine $4-$cycles and one $6 - cycle$. In the following description of π, the cycles are grouped together; for instance the first $4-$cycle has edges $(v_1, u_{10}), (v_9, u_2), (v_1, u_2), (v_9, u_{10})$. The first two of them belong to perfect matching M_3, the last two belong to perfect matching M_2. $[\pi(1) = 10, \pi(9) = 2]; [\pi(2) = 6, \pi(5) = 3]; [\pi(3) = 9, \pi(8) = 4]; [\pi(7) = 13, \pi(12) = 8]; [\pi(14) = 11, \pi(10) = 15]; [\pi(11) = 18, \pi(17) = 12]; [\pi(19) = 14, \pi(13) = 20]; [\pi(16) = 21, \pi(20) = 17]; [\pi(15) = 19, \pi(18) = 16]; [\pi(21) = 7, \pi(4) = 5, \pi(6) = 1]$.

Hence the final sequence (for ease of understanding we preserve the separation symbols ’*’, which in reality would not be present):.

’**ingalgo*r pn inf paape*r is a pis p*ape inthe aipi*s to prof this c*onconcer pn inf*ori foronconce*a cithm present*esetion m of tim*ingalgoa cithm*the aipian ilg*ogorithape in* presentogorith*map aico ent *an ilgesetion *co ent s to pro* pis pmap ai*m of timr is a*f this cmatifo*cealingori for*matifocealing* paape**’

4.3. **Procedure** $S^{1+}(\omega, lb, ub)$**.** If the input of the procedure S^1 comes from several runs of the preparatory procedure S described above, then we need to modify S^1 in order to make its output generic, that is to intentionally preserve the attacker-confusing overlaps. This modified procedure is called S^{1+}.

We recall that S^1 repeats the whole blocks P_i, i.e. the output of S^1 is the cyclic sequence

$$P_1 P_2 P_{\pi(2)} P_{\pi(2)+1} \dots P_{\pi^{-1}(1)-1} P_{\pi^{-1}(1)}.$$

We assume that the input ω of S^{1+} comes from repeated runs of procedure S and so ω contains a lot of segments of length o (the overlaps of runs of S) repeated at least twice; let us denote by R the set of all these segments.

Procedure S^{1+} starts as S^1 by partitioning of ω into blocks

$$P_1, P_2, \dots, P_m.$$

The blocks of S^{1+} cut some of the segments from R. To reflect this, we write $P_i = r^T_{i-1} Q_i r^I_i$ where

- Segment r^T_{i-1} is an empty segment or a terminal segment of an element of R cut by the partition between blocks P_{i-1} and P_i.
- Segment r^I_i is an empty segment or an initial segment of an element of R cut by the partition between blocks P_i and P_{i+1}.

Summarizing this notation we write

$$P_1 P_2 \dots P_m = Q_1 r_1 Q_2 r_2 Q_3 r_3 \dots r_{m-1} Q_m r_m,$$

where each r_i is such an element of R that is cut by the blocks of S^{1+}, or an empty segment. Each $P_i = r^T_{i-1} Q_i r^I_i$ where $r_i = r^I_i r^T_i$.

The first difference of S^1 and S^{1+} is that the overlaps of S^{1+} are not the whole preceding blocks. Instead, the overlap added in front of block P_{i+1} is $Q_i r^I_i$. Hence, block P_{i+1} with the overlap added in front of it has form $Q_i r_i Q_{i+1} r^I_{i+1}$.

To make the cards of S^{1+} more generic (see the same step in the description of Procedure S), we change each such $Q_i r_i Q_{i+1} r^I_{i+1}$ into $Q_i r_i Q_{i+1} r'_{i+1}$ where r'_{i+1} is obtained from r^I_{i+1} by adding a segment so that r'_{i+1} has length o and is repeated elsewhere in ω.

Summarising, the output of S^{1+} has form

$$Q_1 * Q_2 * Q_{\pi(2)} * Q_{\pi(2)+1} * \dots * Q_{\pi^{-1}(1)-1} * Q_{\pi^{-1}(1)} *,$$

where each $*$ stands for a segment of length o which is repeated (at least) twice in this output, or the empty string. More specifically, if $*$ follows segment Q_i then it is equal to r_i or to r'_i.

4.4. **Procedure** $S^2(\omega, o)$**.** Let $S^2(\omega, o)$ be as follows: we assume its input is an output of S^1, i.e. it is the cyclic sequence

$$P_1 P_2 P_{\pi(2)} P_{\pi(2)+1} \dots P_{\pi^{-1}(1)-1} P_{\pi^{-1}(1)}.$$

Note that in this sequence, each block P_i appears twice. Procedure S^2 first *cuts* each P_i randomly into P_i^1, P_i^2 so that length of P_i^1 is at least o, i.e. the whole overlap of length o, which we denote by o_i, is contained in P_i^1. The trick of the concealing algorithm is that *both copies of each P_i are cut in the same way!* Let $o_iP_i^2$ denote P_i^2 with the added overlap.

For example, if P_i is equal to 'abcdefghijkl' and $o = 3$ then a possible cut of S^2 is 'abcde+fghijkl'; P_i^1 is equal to 'abcde', P_i^2 is equal to 'fghijkl' and $o_iP_i^2$ is equal to 'cdefghijkl'.

We may describe the set of the cards of S^2 as the *disjoint* union of two sets $C_1 \cup C_2$, where

$$C_1 = \{o_1P_1^2P_2^1, o_2P_2^2P_3^1, \ldots, o_mP_m^2P_1^1\}$$

and

$$C_2 = \{o_1P_1^2P_{\pi(1)}^1, o_2P_2^2P_{\pi(2)}^1, \ldots, o_mP_m^2P_{\pi(m)}^1\}.$$

We remark here that the cards of C_1 correspond to the edges of perfect matching M_2 of graph $G(\pi)$ and the cards of C_2 correspond to the edges of perfect matching M_3 of $G(\pi)$ (see Definition 4.1).

Finally S^2 arranges $C_1 \cup C_2$ into a random cyclic sequence.

For illustration we perform S^2 on the output sequence of the previous example 2 (which would be the natural use of S^2, as described later):

Example 3: Procedure $S^2(\omega, o)$

Input ω = **'ingalgo*r pn inf paape*r is a pis p*ape inthe aipi*s to prof this c*onconcer pn inf*ori foronconce*a cithm present*esetion m of tim*ingalgoa cithm*the aipian ilg*ogorithape in* presentogorith*map aico ent *an ilgesetion *co ent s to pro* pis pmap ai*m of timr is a*f this cmatifo*cealingori for*matifocealing* paape'**, parameter $o = 3$.
A consistent partitioning into blocks is indicated below:
'inga+lgo*r pn i+nf pa+ape*r is+ a pis+ p*ape+ inthe a+ipi*s to +prof this +c*onco+ncer pn i+nf*ori+ foronco+nce*a ci+thm pre+sent*eseti+on m of +tim*inga+lgoa ci+thm*the a+ipian i+lg*ogori+thape+ in* pre+sentogori+th*map +aico e+nt *an i+lgeseti+on *co e+nt s to +pro* pis+ pmap +ai*m of +timr is+ a*f this +cmati+fo*ceal+ingori+ for*mati+foceal+ing* pa+ape'
Next we add overlap (of length o) in front of each block (and we delete the 'helpful symbol' *):
'apeinga+ngalgor pn i+n inf pa+ paaper is+ is a pis+pis pape+ape inthe a+e aipis to +to prof this +is conco+nconcer pn i+n infori+ori foronco+nconcea ci+ cithm pre+presenteseti+etion m of +of timinga+ngalgoa ci+ cithmthe a+e aipian i+n ilgogori+orithape+ape in pre+presentogori+orithmap +ap aico e+o ent an i+n ilgeseti+etion co e+o ent to +to pro pis+pis pmap +ap aim of +of timr is+ is af this +is cmati+atifoceal+ealingori+ori formati+atifoceal+ealing pa+ paape'
Finally we rearrange the cards in a random order. The resulting sequence is as follows:
'n inforio ent s to ori formati paapen ilgesetipis papen ilgogoringalgor pn iapeingaof timr is is af this presentesetin inf paealingoriealing papresentogorietion m of atifocealap aim of ngalgoa cie aipisan iof timingaatifocealis cmatipis pmap orithapeis concoori foroncoto pro pise aipis to paaper isnconcer pn ietion co e is a pis cithm preo ent an ito prof this nconcea ciap aico eape inthe aorithmap cithmthe aape in pre'

4.5. **Procedure** $S^{2+}(\omega, o)$**.** We assume its input is an output of S^{1+}. This procedure is defined analogously as S^2 with the only difference that the *cuts* are performed to segments Q_i instead of segments P_i.

5. The concealing algorithm

Let the input string be ω, and the length of the preserved segments be k. We consider two scenarios, *weak concealing* and *strong concealing*, depending on the nature of the input. We perform the *weak concealing algorithm* if the input is nonspecific, i.e., short segments have many possible alternative prolongations, or there does not exist any outside knowledge about the likelihood of presence of some segments in the input (e.g. an English text).

The *weak concealing algorithm* may be described as

$$\omega_F = S^2(S^1(\omega, 3k/2, 2k), k-1).$$

We choose to have the block length in S^1 longer and to overlap the whole blocks in S^1 since we want to ensure that the *cuts* of S^2 may be done in the same way in each of the two copies of the blocks P_i.

The *strong concealing algorithm* may be written as

$$\omega_F = S^{2+}(S^{1+}(S \dots S(\omega, k-1, k, 3k/2))), 3k/2, 2k), k-1),$$

where the number of repetitions of procedure S is application specific.

6. Analysis of the concealing algorithm

Observation 6.1. *The concealing algorithm preserves all segments of length k present in the input sequence ω within the output sequence ω_F.*

This observation is straightforward as whenever any of the above procedures cuts the input string, an overlap of length at least $k-1$ is added in front of the segment following the cut, thus preserving all subsegments of length k which would otherwise be partitioned by the cut.

It is also straightforward that both weak and strong concealing algorithms are linear in $|\omega|$ if we have

- Access to a generator of random permutations of the numbers less than $|\omega|$,
- Access to a generator of random elements of $\mathcal{A}$ (see Definition 4.3).

A random permutation may be generated in linear time (see [9]). We will not discuss the complexity of generating random elements of $\mathcal{A}$. Instead, we specify a large subset $\mathcal{B}$ of $\mathcal{A}$ such that generating a random element of $\mathcal{B}$ may be reduced to generating a random permutation of a number less than $|\omega|$.

Each element of $\mathcal{B}$ may be constructed as follows: we take any permutation π of $m/2$ (we assume m is even) and we consider the pairing $P(\pi)$ of $\{1, 2, \dots m\}$ given by $(1, \pi(1)+m/2), \dots (m/2, \pi(m/2)+m/2)$. This pairing may be looked at as an involution $i(\pi)$ (a permutation α is involution if $\alpha(\alpha(x)) = x$ for each x) on m. Finally, we get element $\underline{=}\underline{(}\pi)$ of $\mathcal{B}$ by shifting $i(\pi)$ by 1, i.e., by letting $\underline{(}a) = i(\pi)(a) + 1$ modulo m; we have an additional condition that $O^j(1) \neq 1$ for $j < m$ and $O(a) = \underline{(}a) + 1$. This condition makes sure that the first condition of the definition of the acceptable permutation is satisfied. The following observation is straightforward.

Observation 6.2.

$$|\mathcal{B}| \leq (m/2-1)!$$

Further, the graphs defined by a permutation from $\mathcal{B}$ are disjoint unions of $m/2$ cycles of length 4. Generating a random element from $\mathcal{B}$ is as hard as choosing a random permutation of $m/2$.

The following observation is also straightforward.

Observation 6.3. *The length of the output of each of the procedures applied to input ω is linear in $|\omega|$. For example, for S and S^1 it is $2|\omega|$.*

7. Hardness of the attacker problem

We recall that the attacker problem introduced in Section 3 (see also Proposition 3.1) reads:

How much information about ω can an attacker deduce from ω_F, $|\omega|$, k and the knowledge of the concealing algorithm?

For instance, the attacker can try to get all the overlaps of S^2 since *assuming ω_F has no accidental repeats* these overlaps appear exactly four times in ω_F and no other segment is like that. The attacker may partition ω_F into cards as indicated by all these overlaps. She gets a collection of cards, with $(k-1)$-length segments marked in the beginning and the end of each card. The attacker wants to overlap these marked segments. Depending on whether *ω_F has accidental repeats*, the attacker possibly *cuts* in more places than were the original cards used in the algorithm. Hence, in her collection of cards some overlaps should not have been considered, and some segments have overlaps with more than one other card. These considerations naturally specify the **domino** and **donkey** problems.

In more realistic situation the attacker does not know the correct list of cards of S^2 and hence she needs to choose which 4−repeats to ignore. We may assume that she has some hints as to which overlaps are 'likely' ok. This is the situation we model by the following problem.

Shortest domino row problem (SDRP). Assume we are given a collection of dominoes (domino will mean a rectangle partitioned vertically into two squares, where one is initial and the other one is terminal), and we are also given a graph on the squares. This graph should be interpreted as the graph of hints. We want to put all the dominoes into a row, so that if two consecutive squares are connected by an edge of the graph, we can put one square on top of the other (i.e., identify them). The aim is to make the resulting row as short as possible, i.e. to satisfy as many hints as possible.

Let us define the (de Bruijn-type) graph $G=(V,E)$ where V is the set of all the squares, and E is the set of the dominoes: edge e_i connects the squares of domino Q_i. The following observation is straightforward.

Observation 7.1. *There is a natural bijection between the set of the Euler circuits (eulerian closed walks) of G and the set of all the circular sequences consistent with the overlapping dominoes $Q_1, \ldots, Q_m$.*

Theorem 1. *The SDRP is search-NP-complete.*

Proof. Assume that in the auxiliary graph, there is edge between two squares if they are equal, but not all such edges are there. This is exactly consistent with our interpretation. Now, in the reformulation with the de Bruijn graph and the Euler circuit, this corresponds to the problem that we are given a graph, with some transitions between neighboring edges recommended, and we want to find an Euler circuit with as many recommended transitions as possible. A particular instance is that some transitions are forbidden, and we want to find out whether Euler circuit where all the transitions are allowed exists. This is known to be NP complete ([10]).

We have in fact a *search instance* of this problem: we know that such an Euler circuit exists, and we want to find it. There is a standard trick which shows that the decision problem is polynomial if the search problem is polynomial:

Assume there is a polynomial algorithm A that solves the search version, and let its running time be n^{10}, say. To solve the decision problem, we apply A to an input. It either finds the right Euler circuit and then the answer is YES, or it runs longer than n^{10}, and then the answer is NO. □

In the **donkey problem** we assume that ω_F *has no accidental repeats.* What the attacker gets? There are two versions of the algorithm. Let us first consider the *strong concealing* where the preliminary step is performed.

1. As described above, using the 4−repeats of length $k-1$ of ω_F, the attacker gets the cards of S^{2+}, i.e. $C_1 \cup C_2$, where
$$C_1 = \{o_1 Q_1^2 r_1 Q_2^1, o_2 Q_2^2 r_2 Q_3^1, \ldots, o_m Q_m^2 r_m Q_1^1\}$$
and
$$C_2 = \{o_1 Q_1^2 r_1' Q_{\pi(1)}^1, o_2 Q_2^2 r_2' Q_{\pi(2)}^1, \ldots, o_m Q_m^2 r_m' Q_{\pi(m)}^1\}.$$
2. The attacker also gets each Q_i^1 and each $o_i Q_i^2$ since these are exactly maximal initial and terminal segments of the cards above which are repeated twice in ω_F.
3. By matching the overlaps, the attacker gets each pair $Q_i^1 Q_i^2$ since the overlap o_i in $o_i Q_i^2$ is a terminal segment of Q_i^1 and we may assume that these cannot be misinterpreted.
4. What the attacker gets from the initial applications of procedure S? Each of their overlaps (of length $k-1$) appears at least twice in the input of S^{1+}. Moreover most of the *cuts* of the procedures S are different. Let us recall here that among

these overlaps may be also the dust. Procedures S^{1+} and S^2 *cut* into some of these. Those cut will remain 2-repeats, those not cut may gain repeats. Moreover, S^{1+} introduces dust in the border of each card: this adds 2-repeats of strings of length $k-1$ undistinguishable from the 2-repeats coming from initial procedures S.

In case weak concealing algorithm is applied, the attacker has $1., 2., 3.$ where $Q_i^2 r_i$ and $Q_i^2 r_i'$ are replaced by P_i^2 and Q_i^1 is replaced by P_i^1.

The next proposition summarises the possible types of repeats introduced by the algorithm.

Proposition 7.2. *All the repeats of ω_F generated by the weak or strong concealing algorithm are those described in* $1., 2., 3., 4.$.

Corollary 7.3. *All the useful information for the attacker problem is $|\omega|$, k, and* $1., 2., 3., 4.$.

The information $1., 2., 3.$ may be described by the auxiliary bipartite graph $G(\pi)$ defined in Definition 4.1.

If the weak concealing algorithm is applied, information [4.] does not exist. The attacker problem is thus reduced to the following:

The donkey-decision problem. The input is a bipartite graph G where the vertices in both parts V_1, V_2 are ordered. Let $V_1 = \{u_1, \dots, u_m\}$ and $V_2 = \{v_1, \dots, v_m\}$. Moreover a segment $s(v)$ of length at least $3k/2$ is associated with each element of V_2. The set of the edges of G is formed by a disjoint union of two perfect matchings M_2, M_3. The attacker needs to reconstruct string

$$s(M_2(1))s(M_2(2))\dots s(M_2(m)),$$

where $M_2(i)$ is the vertex of V_2 connected with $u_i \in V_1$ by an edge of M_2.

The difficulty of the donkey-decision problem is the following: bipartite graph G is a union of two edge-disjoint perfect matchings. Each vertex of G thus has degree 2 and G is a union of disjoint cycles. To solve the donkey-decision problem, one needs to *choose the correct perfect matching* independently in each of these cycles (namely, the perfect matching induced by M_2). This is impossible, and the list of all the possibilities is almost always exponential in the number of the cycles, since each of the cycles has two perfect matchings. This is analysed precisely below, when we speak about the *feasible solutions.*

Next we argue that, when the strong concealing algorithm is applied, the attacker problem is reduced to the donkey-decision problem too. The attacker is left with the statistics of the repeats of ω_F. Here comes the reason why we introduced the dust in S^{1+}: it is to make sure that the 2-repeats appear symmetric for both matchings M_2, M_3. This hides the repeats introduced by the initial applications of procedure S. The information of [4.] is thus useless. We obtain:

Proposition 7.4. *The attacker problem for both strong and weak concealing is reduced to the analysis of the donkey-decision problem.*

A *feasible solution* to the donkey-decision problem is any sequence $s(M(1))s(M(2))\dots s(M(m))$, where M is any perfect matching of the input bipartite graph G. In order to solve the donkey-decision problem, one needs to choose, from the pool of these feasible solutions, the unique correct one. Next we argue that unless the input to our problem is extremely restrictive, there is an exponential number of the competative solutions.

The bipartite graphs G coming from $\mathcal{A}$ have at least $2^{m/c}$ perfect matchings. The output sequences of two perfect matchings M, N may still be equal: if the cycle has length 4, this happens if and only if the two vertices v_i, v_j of V_2 in each $4-$cycle in which M, N differ have the same associated segment ($s(v_i) = s(v_j)$ as defined in Observation 4.4).

For instance, if all the vertices of V_2 have the same associated segment, then there is only one competative solution. This extreme situation may happen if the input ω is a sequence of repetitions of one symbol only.

If two symbols may appear in the segments (of length at least $3k/2$) associated with the vertices of V_2, then the probability that in a 4-cycles the corresponding pairs of strings are indistinguishable is $2^{-3ka/2}$. Hence with only exponentially small probability there is less than an exponential number of feasible solutions.

8. Conclusion

We define the information concealing problem and propose an algorithm to solve it. It is based on the intuition coming from the difficulties of DNA reconstruction by hybridisation. The algorithm may be efficiently implemented. In analysing the amount of information leaked by the concealing algorithm to an attacker (this is called the *attacker problem* in the paper), we first consider the case that the output contains random repeats; this leads to the *domino problem* which is shown to be NP-complete. Even if the attacker solves the domino problem, she is faced with the *donkey problem* which is reduced to the *donkey-decision problem*. It is shown that with high probability the donkey-decision problem has an exponential number of feasible solutions among which the attacker needs to choose the correct one.

References

[1] J. Blamey, L. Kencl, M. Loebl, DNA self-concealing by repeats, *Under review*.

[2] R. Ramaswamy, L. Kencl, and G. Iannaccone. Approximate fingerprinting to accelerate pattern matching. In *Proceedings of the ACM Internet Measurement Conference (IMC)*, Rio de Janeiro, Brazil, 2006.

[3] H.-A. Kim and B. Karp. Autograph: Toward automated, distributed worm signature detection. In *Proceedings of the 13th Usenix Security Symposium (Security 2004)*, San Diego, CA, August 2004.

[4] S. Singh, C. Estan, G. Varghese, and S. Savage. Automated worm fingerprinting. In *Proceedings of the ACM/USENIX Symposium on Operating System Design and Implementation (OSDI)*, San Francisco, CA, USA, 2004.

[5] R. Guy-Franck, F. Paques, EMBO Reports 1, 122-126, 2000.

[6] P.A. Pevzner, Computational Molecular Biology: An Algorithmic Approach, *The MIT Press Cambridge, Massachusetts, London, England* (2000).

[7] R. Arratia, B. Bollobas, D. Coppersmith, G.B. Sorkin, Euler circuits and DNA sequencing by hybridization, *Discrete applied Mathematics* 104, 63-96, 2000.

[8] M. Singer, P. Berg, Genes and Genomes, *University Science Books Sausalito, California* (1991).

[9] D. E. Knuth, The Art of Computer Programming (III edition), *Addison-Wesley, Reading, Massachussets* (1997).

[10] M. R. Garrey, D. S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness.

[11] J. Xu, J. Fan, M. H. Ammar, and S. B. Moon. Prefix-preserving IP address anonymization: Measurement-based security evaluation and a new cryptography-based scheme. In *Proceedings of the IEEE International Conference on Network Protocols (ICNP)*, 2002.

[12] D. Maltz, J. Zhan, G. Xie, H. Zhang, G. Hjalmtysson, J. Rexford, and A. Greenberg. Structure preserving anonymization of router configuration data. In *Proceedings of the ACM Internet Measurement Conference (IMC)*, Taormina, Italy, 2004.

[13] R. Pang and V. Paxson. A high-level programming environment for packet trace anonymization and transformation. In *Proceedings of ACM SIGCOMM*, 2003.

[14] B. Bloom. Space/time tradeoffs in hash coding with allowable errors. In *CACM*, page 422.

[15] K. Shanmugasundaram, H. Broennimann, and N. Memon. Payload attribution via hierarchical Bloom filters. In *Proceedings of the ACM Conference on Computer and Communications Security (CCS)*, Washington D.C., USA, October 2004.

[16] Y. Li, J. Tygar, and J. M. Hellerstein. Private matching. In *Proceedings of the 30th VLDB Conference*, Toronto, Canada, 2004.

[17] R. Agrawal, A. Evfimievski, and R. Srikant. Information sharing across private databases. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, San Diego, CA, USA, 2003.

[18] S. Garriss, M. Kaminsky, M. J. Freedman, B. Karp, D. Mazid̀res, and H. Yu. Re: Reliable email. In *Proceedings of the Symposium on Networked Systems Design and Implementation (NSDI)*, San Jose, CA, USA, 2006.

[19] Net 2000 Ltd. Data masking whitepapers.
http://www.datamasker.com/, 2005.

[20] G. Miklau, D. Suciu, A Formal Analysis of Information Disclosure in Data Exchange, *Proc. of ACM SIGMOD Intl. Conf. on Management of Data* (2006).

[21] R. Agrawal, R. Srikant, Privacy-preserving data mining *Proc. of ACM SIGMOD Intl. Conf. on Management of Data* (2000).

[22] S. Agrawal, J. R. Haritsa, A framework for high-accuracy privacy-rpreserving mining, *Proceedings of the 21st International Conference on Data Engineering ICDE* (2005).

[23] A. Evfimievski, J. Gehrke, R. Srikant, Limiting privacy breaches in privacy preserving data mining, *Proc. of ACM Symp. on Principles in Database Systems PODS* (2003).

[24] N. Mishra, M. Sandler, Privacy via pseudorandom sketches, *Proc. of ACM Symp. on Principles in Database Systems PODS* (2006).

[25] V. Rastogi, D. Suciu, S. Hong, The Boundary Between Privacy and Utility in Data Publishing, *Proc. of Intl. Conf. on Very Large data Bases VLDB* (2007).

[26] M. Hay, M. Miklau, G.D. Jensen, S. Weiss, et.al., Anonymizing Social Networks, *University of Massachusets, Amherst, Technical Report* (2007).

[27] N. F. Johnson, Z. Duric, Z. and S. Jajodia, Information Hiding: Steganography and Watermarking - Attacks and Countermeasures, *Springer* (2000).

[28] I. Cox, M. Miller, J. Bloom, J. Fridrich and T. Kalker, Digital Watermarking and Steganography, Second Edition, *The Morgan Kaufmann Series in Multimedia Information and Systems* (2007).

[29] L. Sweeney, k-anonymity: a model for protecting privacy, International Journal on Uncertainty, Fuzziness and Knowledge-based Systems, 10 (5), 557-570, (2002).

[30] C.D. Manning, P. Raghavan and H. Schďtze, Introduction to Information Retrieval, Cambridge University Press, (2008).

[31] L. Kencl, J. Zamora and M. loebl, Packet Content Anonymization by Hiding Words, IEEE INFOCOM, (2006).