Repetition Pattern Attack on Multi-word-containing SecureString 2.0 Objects

Günter Fahrnberger

University of Hagen, North Rhine-Westphalia, Germany
guenter.fahrnberger@fernuni-hagen.de

Abstract. Cloud computing appeals to private individuals and particularly enterprises at a progressive rate, but a noticeable percentage of them refuse it due to mistrust or missing commitment to security. The cryptosystem SecureString 2.0 was designed to outweigh these deficits through the support of blind computations on character strings. Repetition pattern attacks count among the most hazardous enemies of SecureString 2.0 objects because reoccurring ciphergrams within them may reveal linguistic identifying features of the correspondent plaintext. This paper analyzes and compares the success probability of repetition pattern attacks on the following three sorts of SecureString 2.0 objects: single-word-containing ones, multi-word-containing ones with a known number of words plus unknown delimiter positions, and multi-word-containing ones with an unknown number of words plus unknown boundary locations. The latter type is expected to provide the highest privacy.

Keywords: Blind computing, Character string, Character string function, Character string operation, Cloud, Cloud computing, Dictionary Attack, Repetition Pattern, Repetition Pattern Attack, Secure computing, String, String function, String operation.

1 Introduction

These days, private individuals and particularly enterprises have to administer and process huge masses of data. Either they house and compute their binary goods in their private computer centers respectively in their home workstations, or they resort to outsourced virtual environments – well-known as (public) cloud computing. While in-house solutions let their owners know where their data reside, redundancy through geographical replicas and scalability lack or demand expensive expansions. Cloud computing offers the opposite with great flexibility and resilience, but the detention of sensitive data as well as reading or writing activities on them become nontransparent and thus probably dangerous.

The resulting challenge consists in the maximum yield of the advantages of both paradigms accompanied by the eradication or mitigation of the disadvantages. An imaginable approach could focus on costly enhancements of the own physical IT-infrastructure to a private cloud to achieve all benefits. The cheaper way is taken by bringing (convenient) security to a public cloud. Security poses as a coarse term and needs further specification. Privacy, integrity, authenticity and resilience represent intersubjective desirable security goals that shall be maintained at all.

Many scientific disquisitions have proposed ways to sustain data security in outsourced unconfident or semiconfident domains. These ones for secure numeric calculations try to explore feasible homomorphic functions. Most of those ones for secure string computations tackle issues in enciphered databases and keyword search in encrypted documents, mainly by dint of trapdoor functions. Both function types – homomorphic and trapdoor ones – form the heart algorithms of blind computing. Blind computing indicates that a cloud application computes ciphertext data without becoming aware of the meaning of input, output and intermediate results. A comparative overview of state-of-the-art published work about blind computing can be found in [7].

SecureString 1.0 [6] belongs to the kind of cryptosystems for blind computing on nonnumerical data. It bases upon a topical underlying symmetric cryptosystem and polyalphabetical encryption. The ciphering scheme encrypts each n-gram (substring of length n) of a word together with the beginning position of the n-gram within the word. Thereby, the start index of each n-gram stipulates the applied alphabet on it. SecureString 1.0 brings the ciphertext n-grams out of sequence after their encryption without losing the possibility to operate on them because their order can be restored after their decryption through their enveloped position information. Due to the finiteness of character string lengths that can be fully supported through querying and replacing operations, every SecureString 1.0 object intendedly contains exactly one word. Disadvantageously, this discloses the string boundaries and abets repetition pattern attacks on them.

Among other improvements, the succeeding SecureString 2.0 [7,8] overcame this limitation to permit an arbitrary number of cohered words per SecureString 2.0 object. In detail, SecureString 2.0 heads for monoalphabetical (substitutional) encryption within each character string, but every utilized alphabet (alphabet = encryption transformation) must not become effective for more than one word. Normally, each encryption transformation depends on a dedicated key, but SecureString 2.0 enciphers each character of the same plaintext character string together with an identical salt (salt = arbitrary nonce). If the salt always transmutes from word to word, then the encryption transformation also changes from word to word, even in case the same key would be employed for all character strings. Automatic salt updating [1,9] is applicable, which means that each salt serves as input for a hash function that outputs the salt for the encryption of the successive plaintext word. This non-size-preserving behavior of SecureString 2.0 entails the advantage that the decryption scheme can simply ignore salts rather than care about them. Optionally for flow control, the salts can be shortened to make room for an appended sequential number that becomes verified during the decryption scheme.

Howsoever SecureString 2.0 ensures a unique encryption transformation for each character string, this treatise examines the first time how the success probability of repetition pattern attacks varies according to the foe's knowledge respectively nescience of the amount of enclosed words in SecureString 2.0 objects whose delimiters (e.g. blanks or full stops) appear as ciphertext characters of ordinary words. The answer for this first inquiry strongly correlates with the upshot of the following second raised question: Does the recognizability of the proper word boundaries, whereupon each of them shares the salt with its left-neighbored character string, abate if an opponent is unaware of the amount of included words in SecureString 2.0 objects? The detectability of the genuine delimiters and hence the success probability of repetition pattern attacks are anticipated to recede if the quantity of ciphered character strings in SecureString 2.0 objects stays unclear for an opponent. The findings of these research questions allow to assess the privacy of SecureString 2.0 objects for various repetition pattern attack scenarios.

Apart from the introductive section, this paper is structured as follows: Section 2 treats of preliminaries that are to be found in related work about repetition pattern attacks. While Section 3 embodies a formal view on repetition pattern attacks on single-word-containing SecureString 2.0 objects, Section 4 does the same for multi-word-containing ones. Ultimately, Section 5 summarizes the obtained examination results and suggests worthwhile future work.

2 Preliminaries

2.1 Repetition Patterns

A repetition pattern characterizes regularities of an observed text through mapping its characters to natural numbers. Appearing equal numbers indicate reoccurring characters. Definition 1 expresses the generator function of a monographic repetition pattern for an arbitrary input text.

Definition 1. *Let Σ be an alphabet, let $v \in \Sigma^l$ be plaintext of length $l \in \mathbb{N}$, and let $v_j \in \Sigma | j \in \mathbb{N} \wedge 1 \leq j \leq l$ be the j^{th} character of v.*

Then $rp(v_j)$ computes the j^{th} number of the repetition pattern of v as follows.

$$rp(v_j) := \begin{cases} 1 & \text{if } j = 1 \\ rp(v_i) & \text{if } (\exists i \in \mathbb{N} | 1 \leq i < j)(v_i = v_j) \\ 1 + max\{(rp(v_1), \cdots, rp(v_{j-1})\}) & \text{if } (\nexists i \in \mathbb{N} | 1 \leq i < j)(v_i = v_j) \end{cases}$$

The complete repetition pattern of v emerges from the ordered set $RP(v) := \{rp(v_j) | 1 \leq j \leq l\}$.

For example, the word "cabbages" causes the repetition pattern "12332456", just as the the character string "guttural" ends in. Evidently, the generator function for repetition patterns is not injective. "Idiomorphs" denote text particles with the same repetition pattern [2]. Cryptanalysts name instantiations of patterns without repetitions as pangrams [5]. The theoretical maximum of distinct repetition patterns with length l equals the count of partitions of a set with

Table 1. Bell numbers

l	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
B(l)	1	1	2	5	15	52	203	877	4140	21147	115975	678570	4213597	27644437	190899322

cardinality l [2], the Bell number $B(l) := \sum_{k=0}^{l-1} \binom{l-1}{k} B(k)$ [3,4]. Table 1 depicts the exponential growth of the Bell numbers. The most recent publication about SecureString 2.0 [8] inadvertently supposed the Catalan numbers as upper limits for producible differing repetition patterns that are growing more slowly than the Bell numbers.

An encryption scheme behaves monoalphabetically if it substitutes each occurrence of a specific plaintext character or of an n-gram with the same ciphertext. This statement even holds for contemporary ciphers with large block sizes (256 bits or more), but the odds (for hackers) look bad that such long plaintext blocks recur. Even if plaintext block repetitions happen, more sophisticated operation modes than ECB (Electronic Code Book) mode [10] impede the use of just one alphabet and therefore ciphertext recurrences. Anyway, if a cryptosystem (such as SecureString 2.0) deliberately processes shorter block lengths (e.g. 7, 8 or 16 bits at monographical substitutional encryption) to maintain the viability of string operations directly on encrypted texts, then repetitive ciphertexts become more probable. For this reason, the incorporated volatile salt in SecureString 2.0 grants ciphertext repeats within character strings only. Nonetheless, recurring plaintext characters or n-grams induce related ciphertext repetitions. The first invariance theorem in [2] confirms that all monoalphabetical substitutions preserve identical repetition patterns between plaintexts and their counterpart ciphertexts. Shannon divides all plain- and ciphertexts into residue classes on the basis of their repetition patterns [14]. In contrast, homophonic and polyalphabetical substitutions as well as transpositions destroy this heredity of repetition patterns [2]. SecureString 2.0 exploits polyalphabetism to avert repeating ciphertext in different words and therefrom deductions of relationships between them.

2.2 Repetition Pattern Attacks

A repetition pattern attack stands out as a subtype of dictionary attacks. To conduct a dictionary attack, an offender requires a repository with an exhaustive list of potentially occurring words [15]. A conventional dictionary attacker tries to break (a secret password of) an authorization mechanism by testing it out with one repository word after another until they reveal the correct element [12]. A repetition pattern attack on a sole ciphertext character string differs from a traditional dictionary offense against an authorization mechanism in two points.

Firstly, only the repetition patterns of a subset of repository words coincide with the repetition pattern of the attacked character string and come into consideration as plaintext candidates. One could assume that a dictionary comprehends $|\Sigma|^l$ words with length l and thus $|\Sigma|$ times more than such ones with with length $l - 1$ (corresponding to $\frac{|\Sigma|^l}{|\Sigma|^{l-1}} = |\Sigma|$). Additionally, it is assumable that the repetition patterns of the $|\Sigma|^l$ character strings with length l split into the $B(l)$ maximally possible alternatives equably. Both assumptions seldom persist in practice. Already the recent survey about SecureString 2.0 [8] remarked that the frequency scale of natural language words with various lengths resembles almost normal distribution rather than equipartition. Also, only the minority of the potential $B(l)$ repetition patterns per length l matches for the vocables of a natural language. The extremely rare incidence of more than two identical letters in a row describes one obvious cause therefor.

Secondly, there exists no primitive deciding algorithm to unambiguously determine the correct candidate. In case of natural languages, programs can decide on orthographical and grammatical mistakes at the best rather than on the meaningfulness of sentences or texts. Computers can combinatorially assemble all candidate combinations and perform rudimentary preselections, but only humans possess sufficient feel for language to can make the final decision on useful text constructs. In case of repetition pattern attacks on character strings that do not appertain to a natural language, solely the evaluation of the most frequently occurrent candidate could help.

If an assailant expects certain words or phrases (ideally such ones without known idiomorphs) in a monoalphabetically enciphered cryptotext, then they can look there for the appearance of the accordant repetition pattern and save a lot of time instead of attempting to link all ciphered character strings to appropriate candidates in the dictionary. Each such discovered cryptotext fragment together with the probable word forms a "crib" [2]. The prosperous detection of cribs claims the normalization of all search positions in the ciphertext as detailed in example 1.

Example 1. Let Σ be an alphabet, let $v :=$ "Hide the bomb near the pump" be the targeted plaintext and $w \in \Sigma^{27}$ its monographically monoalphabetically encrypted counterpart, let $u :=$ "bomb" be the sought keyword and $q \in \Sigma^4$ its monographically monoalphabetically enciphered counterpart, then the function in Definition 1 generates the same repetition pattern $RP(v) = RP(w) :=$ "1 2 3 4 ␣ 6 7 4 ␣ 8 9 10 8 ␣ 11 4 12 13 ␣ 6 7 4 ␣ 14 15 10 14" for v and w respectively the same repetition pattern $RP(u) = RP(q) :=$ "1 2 3 1" for u and q.

Subsequently, $RP(q)$ must be compared with each repetition pattern of all six embraced character strings of w as follows: "1 2 3 4", "1 2 3", **"1 2 3 1"**, "1 2 3 4", "1 2 3" and **"1 2 3 1"**. The bold repetition patterns label the two idiomorphic substrings "bomb" and "pump". The keyword "bomb" poses as a suboptimal example due to a bulk of coexisting idiomorphic English expressions for the repetition pattern $RP(q)$, but quests for words with inimitable repetition patterns would promise better success.

Even if the spaces in v would become ciphered monograms in w rather than staying invariant and thence exposed, the comparison of $RP(q)$ with each of the following 24 repetition patterns (which reflect all four characters long substrings of w) identifies "bomb" and "pump": "1 2 3 4", "1 2 3 4", "1 2 3 4", "1 2 3 4", "1 2 3 4", "1 2 3 4", "1 2 3 4", "1 2 3 4", "1 2 3 4", **"1 2 3 1"**, "1 2 3 4", "1 2 3 4", "1 2 3 4", "1 2 3 4", "1 2 3 4", "1 2 3 4", "1 2 3 4", "1 2 3 4", "1 2 3 4", "1 2 3 4", "1 2 3 4", "1 2 3 4", "1 2 3 4" and **"1 2 3 1"**.

(Especially long) terms with unique repetition patterns can also be spotted through those of their subpatterns that are unique as well. Queries for shorter words respectively their resultant repetition patterns skimp on time. For the same purpose, ciphergram look-ups for the nominative case of nouns respectively for infinitive verbs may take place before continuing with their declined or conjugated variants.

3 Repetition Pattern Attack on Single-word-containing SecureString 2.0 Objects

In case of the SecureString 2.0 cryptosystem, the most easily breakable ciphertext accrues if an opponent has the knowledge that it only incorporates such SecureString 2.0 objects whereupon each of them comprises of exactly one encrypted dictionary entry. If a plaintext includes a dictionary word $v_m | m, o \in \mathbb{N} \wedge 1 \leq m \leq o$ with an unrivaled repetition pattern, and a SecureString 2.0 object merely consists of v_m's ciphered form w_m in accordance with Theorem 1, then $|U_m| = 1$ and thence $p(v_m) = p(w_m) = \frac{1}{1} = 1 \Rightarrow$, i.e. a repetition attack can uncover w_m very easily. Consequently, the probability of exposing one character string of w attains the maximum value 1 with $1 - \prod_{k:=1}^{o} 1 - p(w_k) = 1 - 0 = 1$ because the factor $1 - p(w_m) = 1 - 1 = 0$ zeros the entire product $\prod_{k:=1}^{o} 1 - p(w_k)$.

Theorem 1. *Let Σ be an alphabet, let $v \in \Sigma^*$ be a plaintext with $o \in \mathbb{N} | o > 0$ delimited character strings, let $v_k \in \Sigma^* | k \in \mathbb{N} \wedge 1 \leq k \leq o$ be the k^{th} word of v and $w_k \in \Sigma^*$ its correspondent SecureString 2.0 object, let $w := \{w_k | 1 \leq k \leq o\}$ be the ordered set of all SecureString 2.0 objects (each with an enciphered word of v), let the function in Definition 1 generate the correct repetition pattern $RP(v_k) = RP(w_k)$ for each v_k respectively w_k, let $Q \subseteq \Sigma^*$ be a dictionary with at least all o words that occur in v, and let $\Omega_k \subseteq Q$ be the probability space with all events (dictionary words) that precisely own the repetition pattern $RP(w_k)$.*

1. *Then*

$$p(w_k) := \frac{1}{|\Omega_k|} | (\forall u \in \Omega_k)(RP(w_k) = RP(u))$$

is the probability of choosing the right dictionary item for w_k,

2.

$$1 - \prod_{k:=1}^{o} 1 - p(w_k)$$

is the probability of guessing the true dictionary word for at least one character string of w, *3. and*

$$p(w) := \prod_{k:=1}^{o} p(w_k)$$

is the probability of choosing the right dictionary item for each character string of w.

Proof:

1. w_k encapsulates only a single event. Therefore, the set $A_k \in \mathcal{P}(\Omega_k)$ with the correct events must be singleton and incloses just v_k. The incidence rate $p(A_k) = p(v_k) = p(w_k)$ can differ from text to text and hence is presumed equally as that of the other members of Ω_k with $p(u)|(\forall u \in \Omega_k)(u \neq v_k)$. If all events in a finite probability space resemble uniform distribution, then the incidence rate for each of them is $p(u) = p(A_k) = p(v_k) = p(w_k) = \frac{|A_k|}{|\Omega_k|} = \frac{1}{|\Omega_k|}$ referred to Laplace's formula.
2. On the contrary, the probability of selecting not the correct event in Ω_k arises from the probability of picking one of the mutually exclusive, complementary events $p(\Omega_k \backslash A_k) := 1 - p(A_k)$. The probability of opting the wrong event for each element of w results from the product of the individual converse incidence rates $\prod_{k:=1}^{o} 1 - p(w_k)$. The converse probability of this product (1 - $\prod_{k:=1}^{o} 1 - p(w_k)$) shows the chances of choosing not all events wrongly, i.e. at least one event correctly.
3. The probability of guessing the proper event for each element of w is the product of the individual incidence rates $\prod_{k:=1}^{o} p(w_k)$. □

A SecureString 2.0 object, which embraces all o units of w with recognizable boundaries, implies just a single data structure with the equivalent vulnerability magnitude as o SecureString 2.0 objects, each with one item of w. The successional subsection investigates the implications of wrapping all elements of w in one SecureString 2.0 object while keeping their boundaries covertly.

4 Repetition Pattern Attack on Multi-word-containing SecureString 2.0 Objects

Malicious parties that snap up a multi-word-containing SecureString 2.0 object with concealed character string delimiters interest themselves in the recognition of these boundaries in order to seek dictionary words with the repetition patterns of the segregated words. Foremost, an evildoer must designate all possible positions of a targeted repetition pattern that suit as delimiter candidates. If multi-word-containing SecureString 2.0 objects do not imbed additional delimiters or miscellaneous characters to create intentional confusion for villains, then Theorem 2 itemizes the characteristics of potential word boundary candidates in repetition patterns.

Theorem 2. *Let Σ be an alphabet, let w be a SecureString 2.0 object that secures a plaintext $v \in \Sigma^l$ without additional confusing delimiters or miscellaneous* *characters, whereupon each word boundary shares the salt with its left-neighbored character string, let $w_j \in \Sigma | j \in \mathbb{N} \wedge 1 \leq j \leq l$ be the j^{th} encased character in w and $rp(w_j)$ its number in the repetition pattern $RP(w)$ agreeable to Definition 1.*

Then each $rp(w_j)$ represents a potential word delimiter candidate if it fulfills all of the following requirements.

1. $1 < rp(w_j) < l$
2. $(\forall i \in \mathbb{N} | 1 \leq i < j)(rp(w_i) < rp(w_j))$
3. $(\forall i \in \mathbb{N} | j < i \leq l)(rp(w_i) > rp(w_j))$

Proof:

1. Supposedly, if the j^{th} encased character of w tags a word delimiter with $rp(w_j) = 1$ respectively $rp(w_j) = l$, then $j = 1$ respectively $j = l$ in agreement with Definition 1. The first respectively the last character of w as word delimiter contradicts the presumption that w does not embed additional confusing delimiters or miscellaneous characters. On account of this, $1 < rp(w_j) < l$ must be valid for a word delimiter candidate w_j.
2. Assuming that the j^{th} encased character of w marks a word delimiter and an $i | 1 \leq i < j$ with $rp(w_i) \geq rp(w_j)$ exists, then at least one $f \in \mathbb{N} | 1 \leq f \leq i$ with $rp(w_f) = rp(w_j)$ must occur in $RP(w)$, because $(\forall rp(h) \in \mathbb{N} | rp(f) < rp(h) \leq rp(i))(\exists g | rp(g) + 1 = rp(h)$ in compliance with Definition 1. The validity of $rp(w_f) = rp(w_j)$ contravenes the premise that only a non-repetitive number in a repetition pattern can flag a word boundary, because each delimiter shares the salt with its left-neighbored character string and each word exhausts a unique salt. On that account, a word boundary candidate w_j demands that $(\forall i \in \mathbb{N} | 1 \leq i < j)(rp(w_i) < rp(w_j))$.
3. Granted that the j^{th} encased character of w typifies a word delimiter and an $i | j < i \leq l$ with $rp(w_i) \leq rp(w_j)$ exists, then at least one $f \in \mathbb{N} | 1 \leq f \leq j$ with $rp(w_f) = rp(w_i)$ must appear in $RP(w)$, because $(\forall rp(h) \in \mathbb{N} | rp(f) < rp(h) \leq rp(j))(\exists g | rp(g) + 1 = rp(h)$ pursuant to Definition 1. Plural existences of the same ciphertext character connote that they adhere to the same salt. Therefrom, $rp(w_f)$ and $rp(w_i)$ and all interjacent enciphered characters belong to the identical word. Accordingly, the also-intermediary $rp(w_j)$ is surrounded by characters of the same character string and cannot act as word boundary anymore. That is why $(\forall i \in \mathbb{N} | j < i \leq l)(rp(w_i) > rp(w_j))$ must have validness for a character string delimiter w_j. □

There may be void combinations of word delimiter candidates. In particular, those ones are invalid in which adjacent candidates deny character strings betwixt them. For this reason, upon ascertainment of the potential word boundaries, all valid combinations of them must be descried in order to calculate the success probability of a repetition pattern attack on each of them and to derive an overall success probability.

4.1 Repetition Pattern Attack on a SecureString 2.0 Object That Contains a Known Number of Multiple Words

If a miscreant becomes aware how many separate character strings a targeted SecureString 2.0 object protects and reckons that exactly one delimiter symbol divides contiguous words, they can stint expenses and continue only with such word delimiter combinations that contain one item fewer than the set of separated words. The number of scrutinized boundary combinations ensues from the amount of all potential delimiters $|D|$ and the quantity of o character strings in a SecureString 2.0 object. Theorem 3 displays the case for a SecureString 2.0 object without a punctuation mark as the last character, i.e. one with o words and $o-1$ boundaries. Furthermore, the Theorem excludes all combinations with adjoining delimiter candidates.

Theorem 3. *Let Σ be an alphabet, let w be a SecureString 2.0 object that secures all $o \in \mathbb{N} | o > 0$ delimited character strings of a plaintext $v \in \Sigma^*$, let $D \subsetneq RP(w)$ be the ordered set of all potential word boundary candidates subject to the outcome of Theorem 2, let $D_m \subseteq D | m \in \mathbb{N} \wedge 1 \leq m \leq \binom{|D|}{o-1}$ be the m^{th} combination of $o-1$ character string delimiter candidates out of D, and let $b_m \in \{1, \infty\}$ be a dichotomous flag that indicates if D_m constitutes a valid word boundary combination.*

Then

$$b_m := \begin{cases} 1 & \text{if } (\nexists d, e \in D_m)(|d-e| = 1) \\ \infty & \text{otherwise} \end{cases}$$

Proof: Due to the familiar fact of o shielded words inside w, an iniquitous cryptographer may presume $o-1$ delimiter symbols that divide them. The binomial coefficient determined by $|D|$ and $o-1$ gives the number of subsets of D with $o-1$ elements. All of these subsets with conterminal boundaries, i.e. such ones with an arithmetic difference of 1, are invalid and hence their binary marks set to infinite. The flags of unobjected combinations are assigned to 1. □

Eventually, a wretch executes a repetition pattern attack in virtue of Example 1 for the o words of each valid boundary combination. Theorem 4 pursues Theorem 3 and outlines his overall success probability of breaking w with it.

Theorem 4. *Let Q be defined as in Theorem 1, let $\Sigma, v, o, w, D, m, D_m, b_m$ be defined as in Theorem 3, let $w_{k,m} \in \Sigma^* | k \in \mathbb{N} \wedge 1 \leq k \leq o$ be the k^{th} encapsulated word in w if the elements of D_m act as character string delimiters in w, and let $p(w_{k,m})$ be the probability of choosing the right item for $w_{k,m}$ in Q.*

Then

$$p(w) := \frac{1}{\sum_{m:=1}^{\binom{|D|}{o-1}} \frac{1}{b_m * \prod_{k:=1}^{|D_m|+1} p(w_{k,m})}} = \frac{1}{\sum_{m:=1}^{\binom{|D|}{o-1}} \frac{1}{b_m * \prod_{k:=1}^{o} p(w_{k,m})}}$$

is the probability of revealing the right word delimiters and the correct dictionary item for each character string of w.

Proof: As proved in Theorem 1, the probability of tipping the right event for each element of w is the product of the individual incidence rates $\prod_{k:=1}^{o} p(w_{k,w})$ if the elements of D_m act as character string delimiters in w. The multiplicative inverse $\frac{1}{b_m * \prod_{k:=1}^{o} p(w_{k,m})}$ counts the quantity of suitable dictionary word combinations. If D_m encompasses an invalid word boundary combination, then Theorem 3 returns $b_m := \infty$ by what the amount of fitting dictionary word combinations becomes zeroed. Each valid combination of character string delimiters contributes such a quantity of matching dictionary word combinations. As a result, all these quantities must be summated to a total amount of suited dictionary word combinations $\sum_{m:=1}^{\binom{|D|}{o-1}} \frac{1}{b_m * \prod_{k:=1}^{o} p(w_{k,m})}$. The reciprocal value $\frac{1}{\sum_{m:=1}^{\binom{|D|}{o-1}} \frac{1}{b_m * \prod_{k:=1}^{o} p(w_{k,m})}}$ is the probability of culling the right character string boundaries and the correct dictionary item for each word of w. $\square$

Despite a criminal's lore about o for a multi-word-containing SecureString 2.0 object, they need to look up the repetition patterns of $\sum_{m:=1}^{\binom{|D|}{o-1}} b_m * o$ ciphertext words instead of o look-ups in summary for o comparable single-word-containing SecureString 2.0 objects. On these grounds, the usage of one o-word-containing SecureString 2.0 object in place of o single-word-containing SecureString 2.0 objects improves privacy rather than abides it.

4.2 Repetition Pattern Attack on a SecureString 2.0 Object That Contains an Unknown Number of Multiple Words

This subsection deals with the success probability of repetition pattern attacks on SecureString 2.0 objects which conceal their amounts and positions of protected words and boundaries. For that reason, a felon must check all subsets of the list with the potential word delimiter candidates D for validity rather than only those with $o-1$ items as demonstrated in Theorem 3. Theorem 5 copies the principle from Theorem 3 how to differentiate between valid and void boundary combinations, but significantly more subsets than in Theorem 3 can incur that need to be processed.

Theorem 5. *Let Σ, v, o, w, D be defined as in Theorem 3, let $D_m \subseteq D | m \in \mathbb{N} \wedge 1 \leq m \leq 2^{|D|}$ be the m^{th} combination of string delimiter candidates out of D, whereby each of the $|D|$ bits of the binary representation of $m-1$ decides the incorporation of a particular candidate of D in D_m, and let $b_m \in \{1, \infty\}$ be a dichotomous flag that indicates if D_m constitutes a valid word boundary combination.*

Then

$$b_m := \begin{cases} 1 & \textit{if } (\nexists d, e \in D_m)(|d-e| = 1) \\ \infty & \textit{otherwise} \end{cases}$$

Proof: In addition to the proof of Theorem 3, it needs an evidence that an m for each subset of D exists, so that D_m equates the subset. $\mathcal{P}(D)$ is the power set of D and enfolds all $2^{|D|}$ diverse subsets of D (inclusive the empty subset with nary a potential character string delimiter candidate and D with all potential word boundary candidates). Let m be stored in an array with as many binary elements as candidates in D, whereupon each bit of m determines the presence of a dedicated candidate in D_m. A bit can adopt one of $2^1 = 2$ states. An array with $|D|$ bits can adopt one of $2^{|D|}$ states. Accordingly, m possesses barely enough states to enumerate all $2^{|D|}$ items of $\mathcal{P}(D)$. $\square$

Finally, a delinquent can move on to execute a repetition pattern attack on the character strings of all valid word delimiter combinations as shown in Theorem 6.

Theorem 6. *Let Q be defined as in Theorem 1, let Σ, v, o, w, D be defined as in Theorem 3, let m, D_m, b_m be defined as in Theorem 5, let $\Delta(b_m, 0)$ be the Hamming weight [11,13] of b_m, i.e. the number of ones in b_m, let $w_{k,m} \in \Sigma^* | k \in \mathbb{N} \wedge 1 \leq k \leq \Delta(b_m, 0) + 1$ be the k^{th} encapsulated word in w if the elements of D_m act as character string delimiters in w, and let $p(w_{k,m})$ be the probability of choosing the right item for $w_{k,m}$ in Q.*

Then

$$p(w) := \frac{1}{\sum_{m:=1}^{2^{|D|}} \frac{1}{b_m * \prod_{k:=1}^{|D_m|+1} p(w_{k,m})}} = \frac{1}{\sum_{m:=1}^{2^{|D|}} \frac{1}{b_m * \prod_{k:=1}^{\Delta(b_m,0)+1} p(w_{k,m})}}$$

is the probability of revealing the right word delimiters and the correct dictionary item for each character string of w.

Proof: Further to the proof of Theorem 4, the differing upper bounds of the product and of the summation sign require to be proved.

Like in Theorem 3, it is assumed that the last wrapped character in w concludes the closing word rather than being a punctuation mark. Thence, each surmised character string of w is followed by a delimiter symbol except the last word. Thereby, w shields $|D_m| = \Delta(b_m, 0)$ boundaries and $|D_m| + 1 = \Delta(b_m, 0) + 1$ character strings. This explains the product of $|D_m| + 1 = \Delta(b_m, 0) + 1$ word incidence rates.

The upper bound of the summation sign $2^{|D|}$ follows from the number of vetted subsets as per Theorem 5. $\square$

The deficiency of knowing the amounts and positions of safeguarded character strings and boundaries in SecureString 2.0 objects aggravates the odds to break them. In the most secure case, v is a pangram, i.e. in obedience to the first condition of Theorem 2, $|v| - 2 = l - 2$ (all but the first and the last) guarded characters in w qualify for potential word delimiters and entail

$$p(w) := \frac{1}{\sum_{m:=1}^{2^{|v|-2}} \frac{1}{b_m * \prod_{k:=1}^{\Delta(b_m,0)+1} p(w_{k,m})}} = \frac{1}{\sum_{m:=1}^{2^{l-2}} \frac{1}{b_m * \prod_{k:=1}^{\Delta(b_m,0)+1} p(w_{k,m})}}.$$

5 Conclusion

Repetition pattern attacks stick out as the most striking measure to break SecureString 2.0 objects because repetitive ciphertext characters may occur in them. On these grounds, this paper provides an investigation over the success probability of such offensives.

After an introduction of the cryptosystem SecureString 2.0 and its predecessor SecureString 1.0, the preliminaries set out with Definition 1 that specifies and exemplifies how to generate and offend a monographic repetition pattern for an arbitrary input text.

Thereafter, Theorem 1 handles the success probability of repetition pattern attacks on SecureString 2.0 objects if an assaulter knows that they just shelter single words. The privacy of such objects becomes highly vulnerable if their repetition patterns merely mesh with few dictionary elements, or, more seriously, only with a lone one.

If a SecureString 2.0 object allegedly harbors several character strings, then the potential boundaries between them can be figured out with the aid of Theorem 2 which presents the necessary properties that a number of a repetition pattern must have to become a candidate.

The Theorems 3 and 4 clarify the two research questions of Section 1 for a known amount of unknown words and their unknown boundaries in SecureString 2.0 objects. Expectedly, such objects proffer harder recognizability compared to single-word-containing ones.

The Theorems 5 and 6 treat the two research questions of Section 1 in case a thug is even not aware of the number of character strings (and of their delimiters) in SecureString 2.0 objects. Such objects come along with the most difficult detectability in comparison with the other two settings but still do not cope modern privacy needs.

The finding of the last attack scenario notably is that the aim of stronger secrecy could be reached with SecureString 2.0 objects that convert every arbitrary plaintext into pangram ciphertext, i.e. ciphergrams never reoccur. The advancement of SecureString 2.0 to accomplish this goal would be valuable future work.

Acknowledgments. Many thanks to Bettina Baumgartner from the University of Vienna for proofreading this paper!

References

1. Anderson, R.J.: Security engineering - a guide to building dependable distributed systems, 2nd edn. Wiley (2008)
2. Bauer, F.L.: Decrypted Secrets: Methods and Maxims of Cryptology, 4th edn. Springer Publishing Company, Incorporated (2010)
3. Bell, E.T.: Exponential polynomials. Annals of Mathematics 35(2), 258–277 (1934)
4. Bell, E.T.: The iterated exponential integers. Annals of Mathematics 39(3), 539–557 (1938)
5. Eckler, A.R.: Pangram variations. Word Ways 10(1), 17 (1977)
6. Fahrnberger, G.: Computing on encrypted character strings in clouds. In: Hota, C., Srimani, P.K. (eds.) ICDCIT 2013. LNCS, vol. 7753, pp. 244–254. Springer, Heidelberg (2013)

7. Fahrnberger, G.: Securestring 2.0 - a cryptosystem for computing on encrypted character strings in clouds. In: Eichler, G., Gumzej, R. (eds.) Networked Information Systems. Fortschritt-Berichte Reihe 10, vol. 826, pp. 226–240. VDI Düsseldorf (June 2013)
8. Fahrnberger, G.: A second view on securestring 2.0. In: Natarajan, R. (ed.) ICDCIT 2014. LNCS, vol. 8337, pp. 239–250. Springer, Heidelberg (2014)
9. Fahrnberger, G.: Sims: A comprehensive approach for a secure instant messaging sifter. In:2014 13th IEEE International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom) (September 2014)
10. Ferguson, N., Schneier, B.: Practical cryptography. Wiley (2003)
11. Hamming, R.W.: Error detecting and error correcting codes. Bell System Technical Journal 29(2), 147–160 (1950)
12. Pinkas, B., Sander, T.: Securing passwords against dictionary attacks. In: Proceedings of the 9th ACM Conference on Computer and Communications Security, CCS 2002, New York, NY, USA, pp. 161–170 (2002)
13. Reed, I.S.: A class of multiple-error-correcting codes and the decoding scheme. Transactions of the IRE Professional Group on Information Theory 4(4), 38–49 (1954)
14. Shannon, C.E.: Communication theory of secrecy systems. Bell System Technical Journal 28(4), 656–715 (1949)
15. Soroka, E.V., Iracleous, D.P.: Social networks as a platform for distributed dictionary attack. In: Proceedings of the 5th WSEAS International Conference on Communications and Information Technology, CIT 2011, pp. 101–106. World Scientific and Engineering Academy and Society (WSEAS), Stevens Point (2011)