ZetaDnaCripto: Método de criptografia baseado em fitas de DNA.

Yasmmin C. Martins

Instituto Militar de Engenharia (IME)
Rio de Janeiro – RJ – Brasil

nim_asay@hotmail.com

Abstract. *This paper describes a method to encrypt informations using concepts of genetic strands. This method involves procedures that transform the plaintext and uses substitution cipher to compose the base sequence of ciphertext, ready to be synthesized, together with it key, molding a symmetric cipher system.*

Resumo. *Este artigo descreve um método de criptografar informações utilizando conceitos de fitas genéticas. Este método envolve procedimentos que transformam o texto em claro e usam cifra de substituição para compor a sequência de bases do criptograma, pronta para ser sintetizada, junto com sua chave, formando um sistema de cifra simétrico.*

1. Introdução

A criptografia baseada em DNA é um assunto relativamente novo, e surgiu a partir da ideia de se criar computadores genômicos. A computação em DNA é um campo da bioinformática que busca a solução de problemas através da manipulação de sequências de DNA. Como é uma área emergente, ainda estão formulando seus métodos, formalismos, como foi no início da computação convencional eletrônica [ISAIA 2004].

Hoje em dia a segurança de um sistema é essencial, não apenas para uso por governos e militares mas também em transações financeiras e comerciais. Com o avanço da tecnologia e das pesquisas, a criptografia tem ganhado métodos inovadores de cifragem. Como as curvas elípticas, a criptografia quântica, a criptografia por DNA etc.

Este artigo tem o objetivo de apresentar uma forma de criptografia baseada em fitas de DNA, com cifra simétrica, utilizando para as operações uma biblioteca chamada BioJava, construída para lidar com dados e operações relacionadas à análise genômica. Outra técnica utilizada, ao final da encriptação e no começo da decriptação, refere-se ao uso do DNA-Pascal, para fazer operações com as fitas.

Além disso, como aplicação do método proposto, será discutido o processo pelo qual poderá garantir o sigilo e a integridade de dados para os procedimentos de análise forense.

Ao longo do texto, o tipo de dado que está sendo manipulado, as bases que formam a sequência de DNA, será melhor explicado assim como algumas de suas operações; será dada também uma breve explicação sobre a biblioteca BioJava e o método DNA-Pascal;

serão discutidas outras abordagens de criptografia simétrica com DNA; então será apresentado o método proposto e por fim, sua aplicação em análise forense.

2. Computação genômica

Esta nova ciência está baseada em formas de processamento de fitas genéticas de DNA, que por sua vez, são compostas de quatro bases nitrogenadas que são adenina (A), citosina (C), timina (T) e guanina (G). As fitas de DNA possuem cadeia dupla, já as fitas de RNA, possuem uma cadeia única, com presença de outra base, a uracila (A). As combinações possíveis destas bases geram códons (grupo de três bases), estes códons podem codificar um ou mais aminoácidos diferentes, que são responsáveis por formar as proteínas.

Alguns conceitos da computação convencional podem ser abstraídas para o modelo genômico, como a representação de dados, que na primeira é feita através de 0 e 1 e na segunda por meio de 4 letras (A C T e G); na primeira as operações feitas com as informações são executadas por meio de circuitos e componentes eletrônicos, já na computação genômica, como se tratam de moléculas, que estão no núcleo das células, as operações a serem feitas com a fita de DNA, devem ser reações químicas microbiológicas.

Na formalização desta forma de computação, os autores têm abstraído operações como as da computação convencional para similar reações como síntese, hibridização, desnaturação, polimerização, técnicas de separação (para busca dentro de uma sequência, por exemplo).

As principais aplicações da computação em DNA são para aproveitar algumas das principais vantagens da manipulação das sequências, como no caso da implementação das memórias associativas que se refere à capacidade de se codificar um grande volume de informação em uma pequena quantidade de DNA (1 bit por nanômetro cúbico); o fato de poder realizar várias operações biológicas em moléculas de um mesmo tubo-teste ao mesmo tempo. Devido a este grande paralelismo, isso se torna muito vantajoso na busca de soluções para problemas combinatoriais. Em criptografia, estas propriedades se tornaram interessantes, pelo poder de esconder uma mensagem, em fragmentos pequenos que podem conter muita informação.

3. Tecnologias utilizadas

Para efetuar as etapas do método proposto foi utilizada a biblioteca BioJava por conter recursos que tornam mais simples e prática a manipulação de sequências e de seus valores de bases. Para simular as operações de síntese, concatenação e corte de sequências, no fim da encriptação e no início da decriptação, foi utilizado o método DNA-Pascal.

3.1. Biblioteca BioJava

BioJava[1] é um projeto dedicado a fornecer um framework em java que possa processar dados biológicos. Ele provê rotinas estatísticas e analíticas, parsers para leitura e escrita

[1] http://biojava.org/wiki/BioJava:CookBookLegacy

em formatos de arquivos deste domínio como fasta, GenBank, EMBL, UniProt e INSDseq.

Ele possui várias ferramentas de análise, como recursos para verificar similaridade entre sequências como o BLAST. Além de ferramentas para extrair localizações e features (características). Além de métodos para transcrição e tradução de sequências. Além do fato de se poder utilizar matriz de pesos e programação dinâmica, com estrutura também para ontologias e definição e manipulação de estruturas. Possui simulações de mutação e recombinação de acordo com uma determinada frequência que auxilia na implementação de algoritmos genéticos.

3.2. Método DNA-Pascal

De acordo com ISAIA, há 4 métodos de formalismos da computação genômica, em relação às operações e procedimentos para estas, dentre as quais estão o método de splicing, o DNA-Pascal, o de construção e o de filtragem. Foi escolhido para ser abordado neste trabalho o método DNA-Pascal, pois através de algumas operações deste método pôde-se provar que existe uma solução, em tempo polinomial, para o problema NP-Completo 3SAT [LIPTON 1994]. Enquanto os outros possuem dificuldades consideráveis de verificação e implementação.

O DNA-Pascal surgiu da união de algumas das instruções já existentes na linguagem Pascal para a computação convencional, com outras operações e testes aplicáveis à computação genômica. Apesar de o alfabeto utilizado pelas duas abordagens de computação ser diferente, o da genômica é um conjunto representado por {A, C, T, G} e o da convencional é um conjunto representado por {0, 1}, os resultados destas operações independem dos conjuntos a serem utilizados.

Este método trabalha somente com números e arrays inteiros não-negativos, também foi acrescentado um novo tipo de dado chamado multiconjunto finito de palavras. A tabela a seguir mostra as operações especiais e o que elas significam, sendo m uma variável que representa um número natural; x é uma palavra; ^ significa "elevado a", a é uma subpalavra e t, t1 e t2 são multiconjuntos finitos de palavras.

Tabela 1. Operações especiais em DNA-Pascal.

Abreviação	*Nome*	*Instrução*
(UN)	Union (União)	t := t1 U t2
(IN)	Initialization (Inicialização)	t := IN(m), IN(M) = {0,1}^m com m ≥ 0
(AS)	Assignment (Atribuição)	t := t1
(EX)	Extraction (Extração, sem retorno)	t := EX(t1, m, a), EX(t1, m, a) = t1 ∩ ({0, 1}^(m-1) a{0, 1}*), com a ∈ {0, 1} com m≥1
(SX)	Subword Extraction (Extração de palavra)	t := SX(t1, x), SX(t1, x) = t1 ∩ ({0, 1}* x{0, 1}*), com x ∈ {0, 1}*
(EW)	Empty Word (Palavra vazia)	t := {ε}
(RA)	Right Adding (Adição pelo lado direito)	t := t1 . a, t1 . a = {za | z ∈ t1}

(LA)	Left Adding (Adição pelo lado esquerdo)	t := a . t1, a . t1 = {az | z ∈ t1}
(CO)	Concatenation (Concatenação)	t := t1 . t2, t1 . t2 = {xy | x ∈ t1, y ∈ t2}
(RC)	Right Cut (Subtração pelo lado direito)	t := t1/, t1/ = {z/ | z ∈ t1}, com za/ = z para a ∈ {0, 1} e ε/ = ε
(LC)	Left Cut (Subtração pelo lado esquerdo)	t := \t1, \t1 = {\z | z ∈ t1}, com \za = z para a ∈ {0, 1} e \ε = ε
(IS)	Intersection (Interseção)	t := t1 ∩ t2

A próxima tabela possui algumas operações de teste, adicionais às operações acima.

Tabela 2. Verificações de teste auxiliares.

Abreviação	*Nome*	*Instrução*	*É verdade se:*
(SU)	Subset (subconjunto)	t1 ⊆ t2	O multiconjunto t1 contém as palavras de t2
(EM)	Emptiness (vazio)	t = ∅	O multiconjunto t é vazio
(ME)	Membership (é membro de)	x ∈ t	A palavra x pertence ao multiconjunto t

Todas estas operações são correspondentes às operações biológicas previamente apresentadas, o DNA-Pascal já foi utilizado para representar, através destas operações da tabela, o problema *Satisfability* que também é um problema NP-Completo.

4. Métodos de criptografia simétrica baseados em DNA similares

A criptografia simétrica é aquela em que a chave que codifica a mensagem é a mesma usada para a decodificação. Os algoritmos simétricos são mais rápidos que os algoritmos de chave assimétrica, onde há uma chave pública para cifrar e uma chave privada para o receptor autorizado decifrar.

Em Bhoir e Mathangi, é proposto um método bem simples de aplicar criptografia, tranformando mensagem em claro em uma sequência de bases. Este método que se baseia em substituir cada letra pelo número ASCII correspondente em formato decimal e então são agrupados em blocos e encriptados por algum algoritmo conhecido, DES por exemplo. Feito isso, o resultado é codificado para o formato binário, e cada uma das quatro bases equivale a uma combinação possível de pares 0 e 1: A para “00”, T para “01”, G para “10” e C para “11”.

Depois, são setados os primers de cada lado da mensagem, estes primers servem como indicadores de paradas e detecção da mensagem. Isso precisa ser colocado antes da comunicação ocorrer, então a sequência com a informação é adicionada depois das paradas. E por fim, pode ser confinado em microarray ou separado em muitas sequências de DNA. Para decriptar a mensagem, são feitas as operações inversas da encriptação.

Em Terec et Al, são propostos dois métodos de encriptação simétrica por DNA, e um método assimétrico também neste domínio, aqui serão falados brevemente somente os simétricos, devido ao método proposto neste artigo ser simétrico.

O primeiro método simétrico proposto foi implementado na plataforma java simples, e consiste de passos como gerar um índice aleatório, pela classe SecureRandom do pacote Java.Security, sendo este índice gerado de acordo com a limitação de 4 pois são encontradas apenas 4 opções de substituição, A, C, T e G, para posterior tradução. Depois deste passo, a chave a ser gerada deve ter o mesmo comprimento do texto em claro. No caso, este texto é a mensagem traduzida segundo o alfabeto de substituição. O tamanho da chave deve ser um múltiplo de três, então quando enviada uma mensagem longa, o comprimento da chave seria enorme. Por isso, a mensagem é separada em blocos e a cifra encripta e decripta um bloco de cada vez. A cifra de DNA então usará uma fração da mensagem original, e o único modo de implementação foi o ECB (Electronic Code Book).

ECB é o modo mais simples, em que cada bloco de texto em claro encripta um bloco de criptograma. A desvantagem deste modo é que o mesmo texto em claro sempre encriptará para o mesmo criptograma, quando se usa a mesma chave. Esta cifra por DNA comporta-se como uma segunda camada de encriptação, na qual é utilizado um HashMap, que é uma estrutura para armazenar pares chave e valor de forma única. Com isso, foi montada uma nova tabela de associação, onde as chaves são a pontuação ou as letras possíveis e o valor é um trio de formado pelas bases.

Como a tabela considera apenas letras minúsculas na associação, antes de fazer a substituição os blocos de texto em claro foram passados para letras minúsculas. E então o resultado depois da aplicação da cifra de DNA, é uma string com caracteres correspondentes ao alfabeto de DNA, e por fim, a mensagem encriptada final é um array de Bytes formado pela operação XOR entre o criptograma anterior e a chave. Para decriptar, basta fazer as operações reversas, como outro XOR entre o criptograma recebido e a chave. A quebra do que foi obtido em grupos de três, uso do HashMap com os pares valor e chave invertidos e então poderá ser lida a mensagem em claro.

O outro método simétrico proposto foi implementado utilizando a biblioteca BioJava. O primeiro passo deste método consiste em transformar cada caracter da mensagem em claro no seu valor binário segundo a tabela ASCII, após isso, uma outra função obtém destes 8 caracteres a string correspondente depois da substituição de cada par de 0s e 1s, por uma das 4 letras possíveis do alfabeto de DNA, sendo utilizado 00 para “A”, 01 para “C”, 10 para “G” e 11 para “T”.

Então, os caracteres codificados anteriormente são buscados no cromossomo escolhido como chave da sessão no início da comunicação. A mensagem em formato de DNA é separada em grupos de 4 caracteres, e cada grupo que é encontrada nas primeiras posições do cromossomo, guarda-se seu índice num vetor de localizações dos grupos no cromossomo. E usando a ferramenta de leitura de arquivos de sequências em formato FASTA, no BioJava, as sequências correspondentes aos índices em que os grupos foram achados, podem ser guardadas numa lista de objetos.

Na última fase de encriptação, para cada caracter da mensagem um índice aleatório do vetor de índices construído é escolhido. E dessa forma, mesmo que a chave seja a mesma para encriptar a mensagem, podem ser obtidos criptogramas diferentes por causa da aleatoriedade dos índices.

Na decriptação, como a chave é a mesma que a usada na encriptação, cada índice recebido com a mensagem codificada pode ser revertido para os grupos de bases, e cada base possuindo seu equivalente em binário, pode-se obter seu valor como caracter ASCII.

A principal fraqueza deste método é que se o atacante interceptar a mensagem, ele pode decodificar facilmente se ele conhecer a sequência de cromossomos codificantes utilizados como chave de sessão.

5. Método proposto

Com base no estudo das metodologias já propostas pelos autores acima citados, foi construído um algoritmo de criptografia baseado em DNA, de forma simétrica. O algoritmo possui 4 passos para encriptar a mensagem, a saber, gerar a matriz de substituição, gerar a chave, transformar a mensagem em binário e enfim com a matriz, a chave e o texto em claro, gerar o criptograma. Estes passos geram tanto a chave quanto o criptograma em formato de sequência de DNA, foram feitos utilizando a biblioteca BioJava.

Para simular a criação e manipulação dessas sequências, foram utilizadas algumas operações do DNA-Pascal, no fim da encriptação e no início da decriptação. Que falam da concatenação das sequências, a síntese das mesmas e a recuperação.

- Passo 1: Transformar a mensagem em binário

Para formatar os dados da mensagem de forma a respeitar a limitação de vocabulário imposta pelo alfabeto utilizado no DNA, assim como nos algoritmos propostos na seção anterior, foi feito um método para fazer esta transformação, através do método getBytes () do Java. Isso retorna, para cada caracter da mensagem um grupo de 8 bits, correspondente ao código ASCII. Como ilustrado na tabela abaixo.

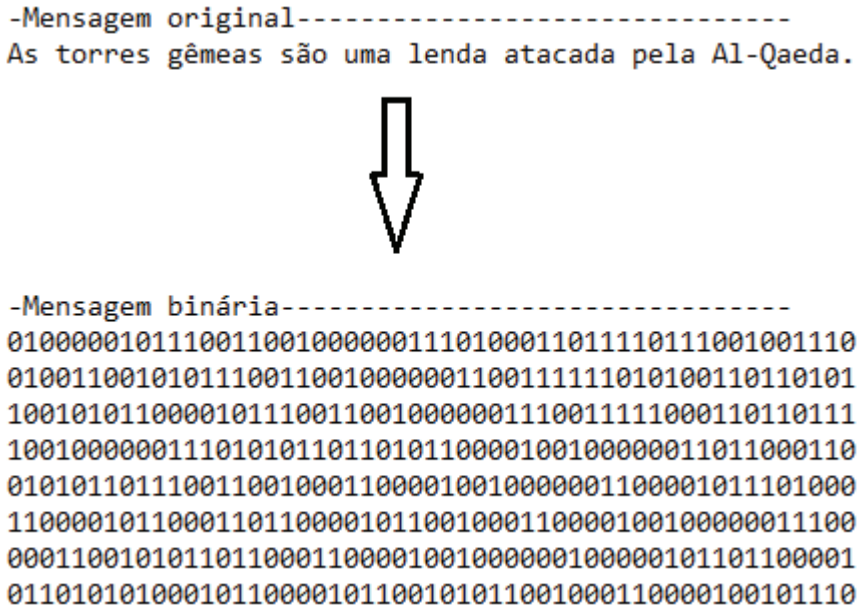

Figura 1. Transformação da mensagem original para binário

Este conteúdo em binário será utilizado pelo próximo passo. Este conteúdo binário é considerado uma String binária, que cresce muito de acordo com o tamanho da mensagem a ser codificada, como pode ser percebido. Uma das vantagens de se usar o DNA para encapsular mensagens é que a compactação de informações neste formato é excelente, possibilitando guardar uma grande quantidade de informações num volume muito pequeno.

- Passo 2: Gerar matriz de substituição

O BioJava entende as sequências como um SymbolList, ou seja, uma lista de símbolos ou uma sequência de bases, os símbolos referem-se a cada base da sequência. Logo, a matriz de substituição nada mais é do que um array de sequências de comprimento 4, que variam seus conteúdos a cada rodada de encriptação. Uma vez gerada, ela é usada para encriptar e decriptar uma mesma mensagem. Logo uma mesma chave, para uma mesma mensagem pode não retornar o mesmo resultado, por causa da aleatoriedade dos conteúdos das posições da matriz. Abaixo, encontra-se um exemplo de matriz gerada.

```
-Matriz-
ctga
actg
gact
tgac
```

Figura 2. Exemplo de matriz de substituição gerada pelo algoritmo

Antes de alimentar a matriz, é feita uma lista de SymbolList, com os valores de sequências possíveis em cada linha. Na hora de preencher esta matriz, os índices da lista auxiliar são embaralhados, e pega-se o que está na primeira posição a cada rodada, e elimina-se a sequência do índice escolhido na lista auxiliar. Assim na primeira vez possui 4 possibilidades, na segunda existem 3 e assim por diante. Pois para fazer o polialfabético funcionar para um mesmo índice de coluna tem que haver elementos distintos nas linhas.

- Passo 3: Gerar chave

Para gerar a chave, para ter mais confiabilidade na aleatoriedade do conteúdo da chave, usando o mecanismo apresentado no segundo algoritmo simétrico proposto em Terec et Al, foi utilizada a funcionalidade de leitura de arquivos de sequências em FASTA, para obter o conteúdo das sequências. Cada arquivo contém sequências contendo acima de 15.000 bases.

Para saber qual arquivo usar e qual o valor de índice a ser pego, usando o SecureRandom, do pacote Java.Security, foram gerados números inteiros aleatórios seguros, dentro do limite de arquivos e dentro do limite do tamanho da sequência existente em cada um, para escolher qual sequência ler e qual o índice do começo da chave. O comprimento da chave escolhido foi de 256.


```
-Chave-----------------------------------------
attgattcatactctatatgttattattttgtatgcgatgacaacagaatatattatcatgctc
cttttgtgaatctcattcataatataaagtataaatttgtgattttgctttaatttgaaatatt
aatttcaaatatgttatcacaatttgatacaaactattgacagtaaatctgtggattaagtaat
gtcttagtaggtattgggaaaatttgaaactagtaacatggaggaatattgtcattgtttattt
```

Figura 3. Chave gerada para o exemplo

A chave acima foi gerada por um arquivo que continha 17.220 bases. O índice não dá para determinar, pode ser qualquer um que esteja entre o valor do comprimento da sequência lida

- Passo 4: Gerar criptograma

Tendo a matriz de substituição, a mensagem em binário e a chave vindos dos métodos anteriores, utiliza-se a matriz para substituir cada par de 0s e 1s, vindo do resultado do passo de transformação da mensagem para binário, pelo índice correspondente da linha da matriz de substituição em que o caracter da chave corresponder ao elemento da 3ª coluna.

Por exemplo, supondo que, para o primeiro par, o valor da mensagem binária é 01, então será substituído por algum símbolo da 2ª coluna da matriz, para saber qual linha que será usada para substituição, pega-se o caracter da chave para esta posição, "a" por exemplo, e compara seu valor com os símbolos da 3ª coluna, no caso a linha escolhida será a 4ª (levando em consideração e matriz mostrada no passo 2), e então este par 01 será substituído por "g". A ordem é 00 para a 1ª coluna, 01 para a 2ª coluna, 10 para a 3ª coluna e 11 para a 4ª coluna. Esta é a lógica utilizada na encriptação.

```
-Criptograma-------------------------------------
ctggacaggttagtgattccgatagcaaccgactaaaaagacgcaaatgtacg
cggcttctaagagaggctgaatcgaagactagtgccgcggatcatattagtga
cgcccagccaccaaagttgcatccggcgaggatgtttggagtaacctaaatta
tcacctcgagtacgtcttattgaatttcctgcgaaacccaacagaaaacttat
```

Figura 4. Criptograma gerado de acordo com as informações anteriores

Com o criptograma e a chave prontos, usa-se as operações do DNA-Pascal, para simular a criação e a concatenação dessas informações numa cadeia de DNA de fato. De acordo com as operações da tabela de operações principal, vista na seção que falava sobre o DNA-Pascal, abaixo está representado um algoritmo básico para terminar a encriptação, com as respectivas explicações das operações.

begin

t1 := IN(m1) => inicializando para o criptograma

t2 := IN(m2) => inicializando para a chave

t := t1.a => Right adding, adicionando um stopper escolhido e combinado com o receptor, para saber onde começa a chave e onde termina o criptograma. Por exemplo, uma sequência fixa de bases.

T_final := t U t2 => Obtida a mensagem final a ser sintetizada por um sequenciador e entregue ao receptor.

end

Para voltar à mensagem original, as operações em DNA-Pascal reversas terão de ser feitas primeiro, a fim de ler separadamente a sequência da chave e a do criptograma. A seguir entra-se outro algoritmo, o de volta. Também com a explicação das respectivas operações utilizadas.

begin

t := IN(m1) => Inicializando com a sequência obtida pelo receptor

t1 := \t => Subtraindo tudo o que estiver do lado esquerdo do stopper escolhido para separar criptograma de chave, obtendo então o criptograma

t_chave := SX(t, x) => Extraindo o stopper (x) do que restou do corte da sequência, obtendo assim a chave.

end

Tendo a matriz de substituição sido previamente revelada ao emissor, e de acordo com as operações acima, lido as sequências do criptograma e da chave, é possível decriptar e mensagem, utilizando agora o BioJava. Com as operações inversas dos passos apresentados anteriormente.

Agora, ao invés de trocar os pares de 0s e 1s para os símbolos do alfabeto do DNA, os símbolos vão ser trocados pelos pares, seguindo a mesma ideia de se o caracter da chave for o mesmo da terceira coluna de alguma linha, esta que foi encontrada servirá de referência em relação aos símbolos que estão armazenados nela, para trocar para o par de 0 e 1 de acordo com o símbolo que está no criptograma.

Com a string binária aplica-se um outro método, para voltar aos caracteres originais da mensagem, ou seja, ler de 8 em 8 bits transformando de volta em char e concatená-los, para obter a mensagem inteira novamente.

6. Aplicação do método em análise forense

Uma das principais etapas de análise forense é a preservação das provas encontradas, ou seja, fazer com que os dados contidos no material que foi levado a exame de peritos sejam inalterados. Para isso é usada a cadeia de custódia que é um documento que registra a história daquela prova cronologicamente. E ela deve assegurar a proteção e idoneidade da prova, a fim de evitar contradições em relação a sua procedência e estado inicial, pois qualquer suspeita pode anulá-la e impactar no processo de investigação [ALMEIDA 2011].

Para garantir a preservação de evidências, alguns métodos discutidos em Almeida envolve a replicação dos dados em uma cópia para que a análise seja feita em cima da cópia; gravação em mídias ópticas; utilização de outro computador caso os dados sejam muito grandes etc.

Algumas das principais preocupações nesta etapa é garantir o sigilo e a integridade dos dados. Para ter certeza de que os dados não tenham sido alterados ou substituídos até concluir o inquérito. Neste ponto, para verificar a integridade utiliza-se alguns mecanismos como o cálculo realizado a partir de funções de autenticação unidirecionais como o hash, que a partir de uma entrada de qualquer tamanho, gera uma

sequência pequena de bits. E ela é muito utilizada para verificação de integridades pois qualquer alteração, por menor que esta seja, gera um hash completamente diferente. Sendo recomendado hashs acima de 128 bits, para não ter colisão e a partir de dois dados diferentes obter o mesmo valor de hash.

Para se ter sigilo pode-se utilizar o método proposto para criptografar tanto os dados quanto o hash, a fim de que sejam preservadas de forma segura e que possam ser validadas novamente pelo hash. Então, como proposto em Schneier, pode-se utilizar também método de criptografia simétrico para criptografar os hashs. E como o método baseado em fitas de DNA, possui uma grande aleatoriedade, os documentos poderão ser mantidos em segurança e intactos.

7. Conclusões

O método proposto necessita da utilização de máquinas específicas, geralmente encontradas em laboratórios modernos, mas foram utilizados procedimentos simples para tais máquinas, significando uma maior velocidade de leitura e descoberta de informação pelas pessoas autorizadas e que mesmo assim dão um nível de segurança muito alto, pois não há uma substituição direta do binário para as bases.

Além disso, a informação é compactada numa escala muito alta, praticamente imperceptível, o que também pode ser considerado como um tipo de esteganografia. Afinal, como o propósito da esteganografia é esconder a existência de uma mensagem, através da síntese de informações criptografadas em fitas que está a nível microscópico, este propósito é alcançado. Além disso, o método proposto agrega uma criptografia antes de obter a sequência.

Outro aspecto deste método é que dentre várias possíveis aplicações ele pode ser de grande valia para a análise forense na preservação do sigilo e da integridade dos dados contidos nas provas, através de seu método de criptografia e esteganografia.

Os avanços nesta área estão sendo concretizados aos poucos, mas sabe-se que pelo grande poder de armazenamento do DNA, as chaves e os criptogramas podem crescer numa escala inimaginável, o que ajuda a dificultar cada vez mais a complexidade de suas gerações.

Referências

ALMEIDA R. N. (2011) “Perícia Forense Computacional: Estudo das técnicas utilizadas para coleta e análise de vestígios digitais”. Trabalho de conclusão de curso, Faculdade de Tecnologia de São Paulo.

BHOIR Y. R.; MATHANGI R. DNA CRYPTOGRAPHY with BINARY STRANDS. https://www.academia.edu/920735/DNA_Cryptography_using_Binary_Strands, Julho.

ISAIA, E. (2004) “Uma Metodologia para Computação com DNA”. Dissertação de mestrado, Universidade Federal do Rio Grande do Sul.

LIPTON, R. J. (1994) “Speeding Up Computations via Molecular Biology”, In: Princeton University, New Jersey. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.50.7037, Julho.

Schneier B. (1996) “Applied Cryptography, Second Edition: Protocols, algorithms, and Source Code in C (cloth)”, In: John Wiley & Sons, Inc., ISBN 0471128457.

TEREC R. et Al. (2011) “DNA Security using Symmetric and Asymmetric Cryptography”, In: International Journal on New Computer Architectures and Their Applications, 1(1), 34-51, ISSN 2220-9085.

