Contents lists available at ScienceDirect

BioSystems

journal homepage: www.elsevier.com/locate/biosystems

DNA-based random number generation in security circuitry

Christy M. Gearheart[a], Benjamin Arazi[b], Eric C. Rouchka[c,*]

[a] Department of Computer Engineering and Computer Science, University of Louisville, Louisville, KY 40292, USA
[b] Department of Electrical and Computer Engineering, Ben-Gurion University, POB 653, Beer-Sheva 84105, Israel
[c] Department of Computer Engineering and Computer Science, Speed School of Engineering, University of Louisville, 123 JB Speed Building, Louisville, KY 40292, USA

ARTICLE INFO

Article history:
Received 31 August 2009
Received in revised form 15 January 2010
Accepted 10 March 2010

Keywords:
DNA-based circuit design
Random number generation
Security circuitry
Oligonucleotide synthesis

ABSTRACT

DNA-based circuit design is an area of research in which traditional silicon-based technologies are replaced by naturally occurring phenomena taken from biochemistry and molecular biology. This research focuses on further developing DNA-based methodologies to mimic digital data manipulation. While exhibiting fundamental principles, this work was done in conjunction with the vision that DNA-based circuitry, when the technology matures, will form the basis for a tamper-proof security module, revolutionizing the meaning and concept of tamper-proofing and possibly preventing it altogether based on accurate scientific observations. A paramount part of such a solution would be self-generation of random numbers. A novel prototype schema employs solid phase synthesis of oligonucleotides for random construction of DNA sequences; temporary storage and retrieval is achieved through plasmid vectors. A discussion of how to evaluate sequence randomness is included, as well as how these techniques are applied to a simulation of the random number generation circuitry. Simulation results show generated sequences successfully pass three selected NIST random number generation tests specified for security applications.

1. Introduction

DNA-based circuit design is an area of research in which traditional silicon-based technologies are replaced by naturally occurring phenomena taken from biochemistry and molecular biology (Fujiwara et al., 2004; Schneider and Hengen, 2004; Seelig et al., 2006). Some experts have hypothesized DNA computers will one day replace their silicon-based counterparts, whereas others believe the future of computing lies in the hybridization of silicon and DNA-based components (Schneider and Hengen, 2004). Fully functional DNA computation can be aided by developing DNA paradigms for converting traditional digital circuitry.

Our team investigates the implications of DNA-based logic circuits in serving security applications, and specifically, building a tamper-proof security module. Current tamper-proof considerations resort to arguments like "it is practically impossible to access the memory from the outside" or "it is impossible to access the data bus that carries the key from storage to the processor if they are all on the same piece of silicon." Technical considerations based on 'good feeling' of engineers make the entire issue of memory security more art than science. It is crucially important to review the entire issue of memory security from a new angle, utilizing new technologies.

An ultimate tamper-proof security module should satisfy three main requirements: resisting static attacks, which involve direct penetration of memory cells where the secret key is stored; resisting dynamic attacks, attempting to retrieve the key as it is passed from memory to the processing element during actual circuit operation; resisting attempts to retrieve the secret key during actual processing. We argue that DNA-based logic circuits, when the technology matures, may provide revolutionary solutions to tamper-proofing. As the gates are based on biological processes, an entire circuit may exhibit features of a combined process, where discrete components, like those observed in CMOS circuitry, are non-existent. Tampering would then have a new meaning, possibly preventing it altogether based on accurate scientific observations. This paper, while presenting the above vision, exhibits initial scientific observations regarding fundamental functioning of a future DNA-based tamper-proof security module.

Since the value to be securely stored is a random secret key, we must first investigate means of generating this value and subsequently storing it. In order to avoid tampering with the key on its way from the generation point to storage, the generation and storage should be in the same place. Furthermore, in terms of com-

* Corresponding author. Present address: Department of Computer Engineering and Computer Science, Speed School of Engineering, University of Louisville, Duthie Center, Room 208, Louisville, KY 40292, USA. Tel.: +1 502 852 1695; fax: +1 502 852 4713.
E-mail addresses: christy.bogard@louisville.edu (C.M. Gearheart), arazi@ee.bgu.ac.il (B. Arazi), eric.rouchka@louisville.edu (E.C. Rouchka).

0303-2647/$ – see front matter © 2010 Elsevier Ireland Ltd. All rights reserved.
doi:10.1016/j.biosystems.2010.03.005

plexity, data storage and retrieval is considered the least difficult. As such, the first research element is to introduce a methodology by which information could reliably be stored and retrieved within a DNA sequence (Bogard et al., 2008). Because of the similarity between a sequence of binary bits and a sequence of DNA characters, a direct substitution table could be used to manipulate the data between the two systems interchangeably. To ensure data is accurately retrieved, multiple sequence alignment enables multiple copies of the same file to find the most probabilistic contents. Like a parity bit, the multiple sequence alignment can indicate that a possible error has occurred. However, while a parity bit can only indicate that an error has occurred, multiple sequence alignment enables the location and type of the possible error to be determined.

Having shown a methodology by which data can be accurately stored and retrieved, the next research component toward devising a DNA-based tamper-proof security module would be to devise a methodology by which one could generate a secret key within the module. As an initial step, a random number generation (RNG) circuitry has been developed. Here, we propose that the secret key, which is actually a random value, be generated by the DNA-based non-volatile memory that subsequently stores the key. A copy of the generated key is then made to share with other friendly parties. Security applications requiring RNG, beside key generation, pertain to nonces (numbers used once), salts in certain signature schemes, and one-time pads. These are essential security components. Any current commercial microchip dedicated to security applications has RNG circuitry and any standard on security applications includes an RNG chapter.

The remainder of the article begins by describing the biological process by which sequences are synthesized in Section 2. Section 3 defines random number generation through DNA sequences. Section 4 describes plasmid vectors, the biological tool by which DNA sequences can be temporarily stored. The random number generation circuitry is discussed in Section 5, followed by the statistical methods utilized to evaluate randomness for the simulation. Finally, justification for DNA-based random number generation is provided in Section 7.

2. Oligonucleotide Synthesis

Deoxyribonucleic acid (DNA) is the molecular material found in the cell's nucleus that encodes an organism's genetic material. DNA is comprised of four nucleotides – adenine (A), cytosine (C), guanine (G), and thymine (T). Just as sequences of binary bits encode possible states within a computer, sequences of nucleotides encode genetic information within an organism. Known as oligonucleotides, these sequences of nucleotides can encode for 4^n possible states, where n represents the length of the sequence. Assuming such sequences can be chemically created, oligonucleotides can be utilized to store information similar to their binary bit counterparts.

Oligonucleotide synthesis is the process in which short sequences of nucleic acids are produced. There are two primary methods of synthesizing oligonucleotide sequences – sequential (Katakai and Goodman, 1982) and solid phase synthesis (Merrifield, 1963). Sequential synthesis occurs by deprotecting the 5′ phosphate then adding the phosphoramidites of the desired nucleic acid in sequential order until the sequence is completed. Sequentially synthesized sequences have a low tolerance to error, and as such are not suitable for creating sequences greater than one hundred nucleotide bases in length.

Solid phase synthesis of an oligonucleotide sequence occurs as a five step process. The 3′ end of the initial nucleotide is bound to a solid support column. A purified solution of the next nucleic acid is then pumped through the support column to adhere a single nucleotide base to the bounded sequence. The remaining solution mixture is then washed out of the support column. The synthesis process continues until the oligonucleotide sequence is created. Finally, the completed oligonucleotide sequence is cleaved from the support column.

Regardless of whether the oligonucleotide sequence is created through sequential or solid phase synthesis, there are four steps to the actual process. The first step, detritylation, releases the 5′ hydroxyl group of the ending nucleotide. Then, the phosphate group of the proceeding nucleotide is removed, enabling the two nucleotides to be bound together. Capping blocks non-reacting nucleotides from incorrectly synthesizing to the sequence, allowing excess nucleotides to be washed off. Finally, oxidation allows the two bounded nucleotides to become permanently stable.

3. Random Number Generation with DNA

A random number, in its primitive form, is a sequence of digits selected at random to generate a number within a given range modeling a given distribution. For example, to generate a random binary number with a range of 0 to 2^{10} following a uniform distribution, one would randomly select either a zero or one independently for each of the 10 bits, with the probability of selecting zero equal to 50% and the probability of selecting one equal to 50%.

To generate a random DNA sequence following a uniform distribution, one would randomly select one of four possible characters – A, C, G, T – for each place in the sequence, with each character having the probability of being selected equal to 25%. Assigning a 2-bit value to each character, a sequence of $4n$ characters generates a random number of n bytes.

3.1. *Physically Synthesizing the Random Number Sequence*

While either method of oligonucleotide synthesis will enable a sequence to be generated, solid phase synthesis is the most effective method of creating a random oligonucleotide sequence. The practical application of randomly assigning a nucleotide to the sequence will simplify the solid phase synthesis process. Rather than using a purified solution of a single nucleic acid mixture, a mixture of nucleic acids of a predetermined distribution could be repeatedly washed through the support column. For example, if a sequence with uniform distribution of each of the four nucleotides is desired, a mixture containing 25% As, 25% Cs, 25% Gs, and 25% Ts can be created. This solution mixture would be continuously used, enabling nucleotides to randomly adhere to the sequence until the desired length is achieved.

It is important to note the simplification of the cleansing process. Solid phase synthesis requires that one must cleanse the support column of any residue nucleic acid to prevent one from erroneously adhering to the sequence. There is no restriction on which nucleotide should adhere to the sequence next, therefore cleansing residue nucleotides from the support column is not necessary since all nucleic acid assignments are valid assignments.

4. Temporary Storage of Random Numbers

Plasmid vectors are small, circular DNA molecules found in bacteria that enable inserted DNA gene sequences to be transported between various organisms (Klug and Cummings, 2003). In order to encompass the gene sequence, plasmid vectors are spliced open with restriction enzymes so the new sequence can be inserted. A restriction enzyme is a small protein sequence that aligns with a specific complementary DNA sequence and cleaves the sequence at such location (Klug and Cummings, 2003).

Rather than inserting a DNA gene sequence to be inserted in a target organism, one can temporarily store a random number by

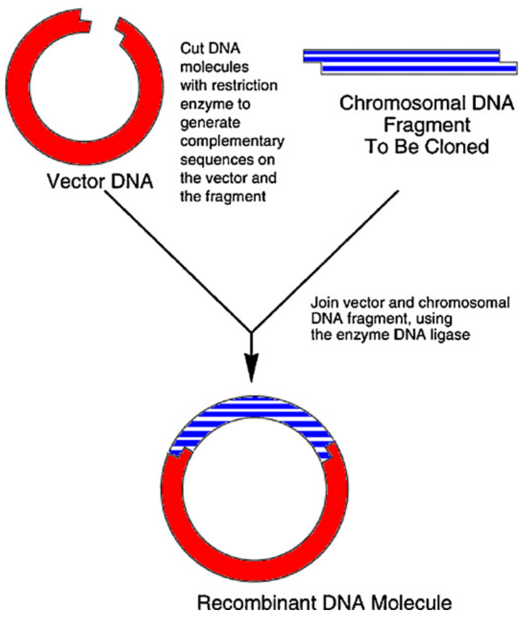
Cut DNA molecules with restriction enzyme to generate complementary sequences on the vector and the fragment
Vector DNA
Chromosomal DNA Fragment To Be Cloned
Join vector and chromosomal DNA fragment, using the enzyme DNA ligase
Recombinant DNA Molecule

Fig. 1. Illustration of the insertion of chromosomal DNA into a plasmid vector cut by a restriction enzyme. Image adapted from U.S. Department of Energy Genome Research Projects (2008).

inserting its corresponding DNA sequence into the plasmid (Fig. 1). The random sequence location is determined by the site selection of the restriction enzyme. The restriction enzyme cleaves the vector open, the random oligonucleotide sequence is inserted, and then the vector is reconstructed to its original circular molecule. In order to retrieve the random sequence from the vector, the process of insertion is reversed.

Once again the restriction enzyme is aligned with the vector to cleave the DNA. The next n bases are sequentially read from the vector, where n represents the length of the random oligonucleotide sequence, and finally the vector is reconstructed to its original circular molecule. It is important to note that the retrieval of the enzyme requires three components: (1) the plasmid vector with inserted sequence, (2) the restriction enzyme used to initially insert the random sequence, and (3) the length of the random sequence.

5. Random Number Generation Circuitry

A random number generation circuit must be capable of creating each component required. The circuit must be able to create the random sequence, translate it into the corresponding random number, and output the random number value.

Once the microfluidic device receives an input signal to generate a random number, the first task is to create a random oligonucleotide sequence. Therefore, there must be some renewable mechanism by which each of the four nucleic acids could be selected as a possible next base. It is envisioned such mechanism would be comprised of four fluidic wells each containing a fluorescently labeled pure mixture of one nucleic acid which could be refilled as quantities became diminished.

A transportation tube would independently pull a specified quantity of each nucleic acid and deposit into the mixing chamber. The mixing chamber would combine the four quantities to create the solution mixture. Using solid phase synthesis, the solution mixture would be poured over a support column to create the random sequence until a given length is reached (Fig. 2).

It is important to note the distribution probability dependence on the solution mixture. If the sequence generated is to have equal distribution of the nucleotides over the length of the sequence, then the same solution mixture can be repeatedly poured over the support column. Since each base should have equal probability of being one of the four nucleotides, it is critical that the solution mixture be based on selection with replacement rather than selection without replacement. Without replacing the adhered nucleotide, the probability of the given base being selected decreases with each additional sequence bit added. However, it is important to note that a minute amount of the solution contains an immense amount of each nucleotide. A quantity of 1 μl contains 5×10^{11} molecules (Hart, 1993). Thus, removing one nucleotide will still maintain an overall equal distribution. Therefore, the mixing chamber could combine 1 μl of each nucleotide solution and continuously pour the solution over the column until the desired sequence length is reached.

Once a sequence is created, it must be translated by passing it through a laser that enables each of the fluorescently labeled nucleotide bases to be distinguished in a chromatogram. A chromatogram is a plot of the intensity of each component as a function of time. Thus, for each location in the sequence, one fluorescent

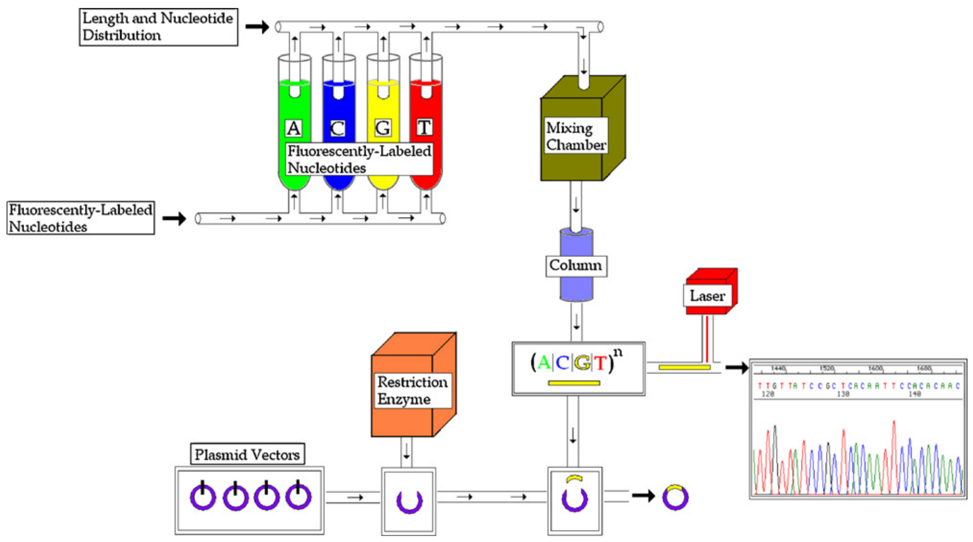
Length and Nucleotide Distribution
A
C
G
T
Fluorescently-Labeled Nucleotides
Fluorescently-Labeled Nucleotides
Mixing Chamber
Column
Laser
(A|C|G|T)^n
Restriction Enzyme
Plasmid Vectors

Fig. 2. Random number generation circuitry. The circuit creates the random oligonucleotide sequence, translates the sequence into its corresponding random number value, and outputs the value in digital form.

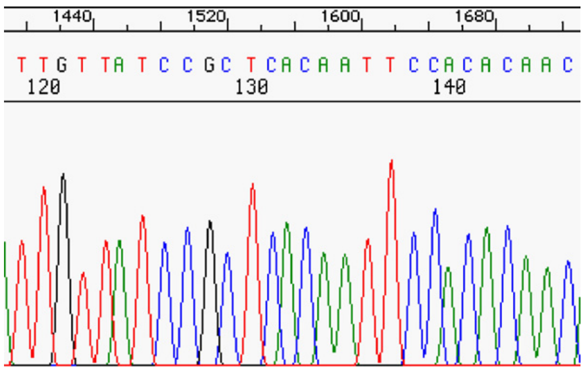

Fig. 3. Chromatogram showing the intensity levels of fluorescently labeled nucleotides for a given oligonucleotide sequence. Image from Lyons (2008).

color will be high intensity while the other three fluorescent colors will be low intensity.

For example, from the chromatogram in Fig. 3, one can see starting at location 120 that the high intensity colors are red, black, red, red, green, red, blue, blue, black, blue, which translates to the nucleotide sequence TGTTATCCGC. Translation from the nucleotide sequence composition to the digitally equivalent random number is achieved through the process described in Section 3.

A created sequence that is not immediately translated quickly becomes deteriorated by environmental factors, making the sequence unusable. Therefore, if a sequence is to be stored for later translation, the circuit must provide a temporary storage mechanism by which the sequence could be preserved. One method of temporary storage involves inserting the sequence into plasmid vectors. Just as the nucleotides were independently pulled from fluidic wells, a plasmid vector could be pulled from an onboard renewable well. Using a restriction enzyme, the vector is spliced open and the random oligonucleotide sequence is inserted to recombine the two spliced ends. The vector has thus encompassed the random sequence into its own DNA, enabling the sequence to be temporarily stored.

Simply creating the sequence and enabling temporary storage is of no value if the sequence cannot be decoded into a digitally equivalent random value. In order to accomplish this, one must first determine the sequence composition in nucleotides. Using the same restriction enzyme used to insert the sequence in the plasmid vector enables one to locate the random sequence in the DNA. After cutting the sequence from the vector, the sequence could then be directly translated. Thus, there are two possible outputs of the microfluidic circuit – (1) the chromatogram of the translated sequence and (2) the plasmid vector temporarily storing the random sequence.

In addition to translating the sequence into its corresponding digital value, it could be beneficial to store the random sequence long term for use at some future time. Rather than outputting the sequence to a laser for translation, one could output the vector-cut sequence to a microarray well location for permanent storage. Thus, one could potentially create a random number repository by generating enough random sequences to fill each location on a microarray, then referencing a new well when a random number is needed.

5.1. *Circuit Fabrication Considerations*

It is essential to evaluate the feasibility of fabricating the circuitry of Fig. 2 as a stand-alone micro-circuit, using current or envisioned future technologies. Size was of crucial consideration in the design of the microfluidic device. Transportation tubes between the various components are on the scale of nanometers. Storage devices are micro-scaled, with a capacity of 10 μl for the various nucleotide solutions, plasmid vectors, and restriction enzymes. As such, fabrication of the device would be on the same scale as their silicon counterparts.

Liquids do not dry up; rather, they are consumed by the circuit just as electricity is consumed by their silicon counterparts. This is not considered a limitation of DNA-based circuitry. Regardless of the venue, there is no perpetual circuit in existence. Just as the silicon chip must be replenished with electricity to remain functional, the DNA chip must be replenished with nucleotide solutions, plasmid vectors, and restriction enzymes.

6. Evaluating Randomness

In order to be truly random, a sequence of numbers must meet two statistical properties – uniformity and independence (Banks et al., 2005; Montgomery, 2001). In other words, every number in the sequence must be selected from a continuous uniform distribution over the interval [0, 1] independent of the selection of any other number. This implies two properties:

1. Every possible value within the interval has an equal probability of being selected as the value of the random variable.
2. Each random number selected is selected completely independent of any previous or future number selections.

A frequency test (Banks et al., 2005; Montgomery, 2001) is used to test the uniformity of a sequence of numbers. A frequency test compares the generated set of numbers to a uniform distribution; the hypotheses are thus:

$$H_0: \quad R_i \sim U[0,1]$$

$$H_1: \quad R_i ! \sim U[0,1]$$

An autocorrelation test (Banks et al., 2005; Montgomery, 2001) is used to test the independence of the sequence numbers. An autocorrelation test compares the correlation between the sequence samples to the expected correlation of zero; the hypotheses are thus:

$$H_0: \quad R_i \sim \textit{independently}$$

$$H_1: \quad R_i ! \sim \textit{independently}$$

For both the frequency and autocorrelation test, one is testing to see if one can reject the null hypothesis (H_0) at a specified level of significance, α. The null hypothesis is rejected when the sequence of numbers shows evidence of being non-uniformity or dependence, respectively. It is important to note that failure to reject the null hypothesis does not directly imply that the sequence is uniform or that the samples are independent; it implies that there is no evidence supporting non-uniformity or dependence using the test at hand. There is no test or set of tests that guarantees that a generated sequence of numbers is truly random.

6.1. *Simulating the Random Number Generation Circuitry*

It is important to simulate the generation of a series of variates to test if the assumptions of uniformity and independence hold, indicating elements that are random. Simulating the proposed random number generator circuit to verify randomness will require a number of tasks; first and foremost is the generation of random variates following uniform distribution.

The simulation is initialized with the number and length of sequences to be generated. Recognizing that a minute amount of DNA solution contains an immense amount of nucleotides, the quantity of each nucleotide available is initially ignored in the initial simulation.

Once initialized, the first step of the RNG Circuitry Simulation is the construction of the nucleotide sequences. To mimic the biological synthesis of nucleotide sequences using solid phase synthesis, each sequence is initialized with a single nucleotide. After all sequences have acquired a single nucleotide, an additional nucleotide is then appended to each of the sequences. This process is continuously repeated until the desired sequence length is achieved for the total number of sequences. While this method of sequence generation is more resource and time intensive, it replicates the solid phase synthesis process utilized in sequential synthesis in a laboratory.

In order to select which nucleotide will be appended to the sequence at hand, the simulation generates a uniform random number between zero and one using a linear congruential random number generator (Knuth, 1997). Because each nucleotide has an equal probability of selection, a piecewise cumulative distribution function will indicate which nucleotide should be selected. In other words, the cumulative distribution function has the piecewise values of 0–0.25 representing A, 0.26–0.50 representing C, 0.51–0.75 representing G, and 0.76–1.00 representing T. The value of the generated random value corresponds to the selected nucleotide to be appended.

Once all nucleotide sequences are created, each sequence is translated to its corresponding binary value through direct substitution. Each nucleotide is sequentially read and the corresponding value is substituted; A, C, G, and T are replaced with 00, 01, 10, and 11, respectively. These translated sequences can subsequently be examined using one of the sixteen standardized analysis techniques of the National Institute of Standards and Technologies (NIST) to test for randomness (Rukhin et al., 2001).

In the ideal setting, each simulation would be tested against all sixteen random number generation tests. Because only a simulation is being tested, the random number generation tests have been limited to three of the most common tests – the frequency (monobit) test, the frequency test within a block, and the runs test (Weigl and Anheier, 2003). These tests were selected because combined, the three tests check for uniformity of values and independence between samples.

The frequency (monobit) test examines the number of zeros and ones present in the entire set of sequences developed. In a truly random system, the proportion of zeros should be equal to the proportion of ones. This test examines how statistically close the number of ones is equal to one-half.

The runs test examines the frequency and length of uninterrupted sequences of identical bits within the entire set of sequences. In other words, this test examines the oscillation between zeros and ones over the entire set of sequences.

The longest runs test is an extension of the runs test that examines the frequencies of the longest run of ones across the sequence set. In other words, the longest run of ones is determined for each sequence, and the overall frequencies are examined to see if they align with the longest run of ones expected in a random sequence of the given length.

In order to gain accurate insight into DNA random number generation, the simulation was run for sequence lengths of 32, 64, 128, 256, and 512 nucleotides, which corresponds to 64-, 128-, 256-, 512-,and 1024-bit sequences (results not shown). For each of these five lengths, 1000, 10,000, and 100,000 sequences were generated and tested for each of the three NIST tests selected. Of all 45 tests run, only two tests failed for the sequences; the two tests that failed were the frequency (monobit) test for 1000 sequences of 256 nucleotides and 1000 sequences of 512 nucleotides. All sequence sets developed passed both the runs test and the longest run test.

Table 1
p-Values of the RNG simulation with nucleotide replacement. *p*-Values less than 0.01 indicates the sequences are not random. All values are greater than 0.01; therefore, all simulated sequences pass the NIST random tests used for analysis.

	p-Value		
	Frequency test	Runs test	Longest run test
32 Nucs			
1k	0.83097	0.34691	0.74026
10k	0.17863	0.79505	0.57023
100k	0.27285	0.80236	0.45020
64 Nucs			
1k	0.16394	0.34203	0.13951
10k	0.45460	0.03576	0.12044
100k	0.08613	0.25331	0.44285
128 Nucs			
1k	0.05830	0.37211	0.26038
10k	0.39602	0.08355	0.35524
100k	0.15820	0.56627	0.53913
256 Nucs			
1k	0.70385	0.16388	0.75932
10k	0.33889	0.46186	0.51952
100k	0.36455	0.25791	0.35728
512 Nucs			
1k	0.90874	0.56119	0.34040
10k	0.93624	0.58318	0.94907
100k	0.80288	0.32967	0.09411

Re-simulating the sets of sequences yields different results (Table 1). This is the direct result of new generated random numbers selecting different nucleotide variates to be appended to the DNA sequence, thereby yielding new sequential values. Re-simulating all fifteen sequence sets results in all 45 NIST random number generation tests being passed.

The simulation confirms that the randomly generated DNA sequences pass three of the NIST tests for randomness. Therefore, it is essential to verify the assumption of nucleotide selection without replacement is valid. The simulation was re-initialized with the number and length of sequences to be generated with the additional variable of the quantity of nucleotides available. By including the quantity of each nucleotide available in the simulation parameters, the assumption of selection without replacement can be scientifically confirmed if the DNA sequences successfully pass the previously confirmed NIST tests for randomness.

In order to include the quantity of nucleotides available in the selection of the nucleotide, the simulation modifies the cumulative distribution function as a function of the percentages of each nucleotide available. For example, if the total nucleotides available is distributed as 23% As, 29% Cs, 21% Gs, and 27% Ts, then the cumulative distribution has the piecewise values of 0–0.23 represents A, 0.24–0.52 represents C, 0.53–0.73 represents G, and 0.74 –1 represents T. Just as before, a random number is then generated; the value of the generated random value corresponds to the selected nucleotide.

Modifying the simulation to generate random DNA sequences without replacement results yields results in which all runs successfully pass all three NIST random number generation tests (results not shown). Assuming that sufficient nucleotides are present to construct all sequences, this confirms that DNA sequence synthesis with nucleotides selected without replacement is in fact a valid assumption.

As an additional independent test of randomness, the melting point temperatures of the nucleotide sequences are examined. The melting point temperature of a sequence is the temperature required to break the bonds between each pair of nucleotides. Melting point temperatures increase in a nonlinear fashion as the length of the nucleotide sequence grows.

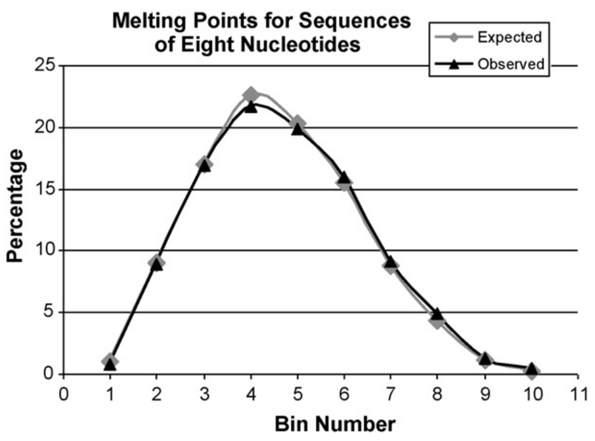

Fig. 4. Expected melting point distribution compared to observed melting point distribution of 10,000 generated sequences of eight nucleotides. Bin numbers represent the equal distribution of the range of all possible melting point values for sequences of eight nucleotides, while the percentages represent the histogram of sequences expected or observed within the given range.

Calculation of the melting point temperature is dependent upon dinucleotide frequencies (Rychlik et al., 1990). For small sequence lengths, one can generate all possible nucleotide sequences, and thus the melting point distribution for the given sequence length. Fig. 4 shows the melting point distribution for all possible sequences of eight nucleotides, indicated by the gray line. Conversely, the black line is the melting point distribution as observed from 10,000 generated sequences of eight nucleotides. Calculating the chi-squared test statistic yields a *p*-value of 0.34242, which is less than the critical test value of 1.152 for 99.9% confidence with nine degrees of freedom, indicating the observed melting points follows the same distribution as expected.

As the length of the sequence grows, it becomes increasing complex to generate all possible sequences and thus the expected melting point distributions. The overall distribution of melting point temperatures is difficult to obtain without generating all possible sequences due to the interdependence of the factors involved. The observed distributions of a large sample set are likely to approach this distribution. Therefore, the observed distributions from 1 million samples were used to test if sample sets of 1000; 10,000; and 100,000 follow the same distribution.

Calculation of the melting point temperature of a nucleotide sequence each sample set of 1000; 10,000, and 100,000 compared to 1 million for sequences of length 32, 64, 128, 256, and 512 nucleotides yields the values summarized in Table 2. All *p*-values are less than the critical test value of 1.152 at 99.9% confidence and nine degrees of freedom, indicating the sample sets follow the same distribution as 1 million samples. The resulting distributions at 1 million sequences are given in Fig. 5.

Table 2
p-Values of sample sets when compared with 1 million samples. All *p*-values are less than the critical test value, indicating the sample sets follow the same distribution as 1 million samples.

	p-Values Critical test value = 1.152 (*p* = 0.001)		
	1k	10k	100k
32 Nucs	0.76069	0.07354	0.00672
64 Nucs	0.19900	0.01897	0.00585
128 Nucs	0.27373	0.04409	0.00496
256 Nucs	0.22612	0.01580	0.00177
512 Nucs	0.01753	0.02765	0.00040

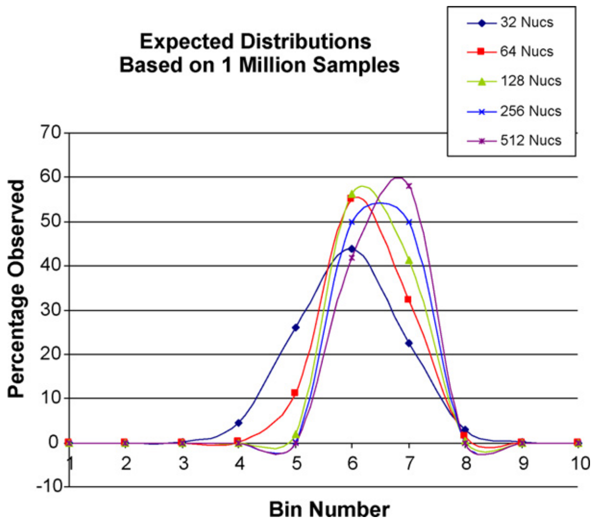

Fig. 5. Expected distributions as calculated from observations of 1 million samples at 32, 64, 128, 256, and 512 nucleotides. Bin numbers represent the equal distribution of the range of all possible sequence values, while the percentage observed represents the histogram of sequences observed within the given range.

7. Justification for DNA-Based Random Number Generation

DNA-based random variate generation using solid phase synthesis enables the development of a myriad of variates in parallel fashion. One cycle of the proposed random number generator circuitry produces approximately 1 million uniformly distributed random variates. However, because of the time constraints required to chemically create and read the oligonucleotide sequences within the lab, it is currently faster to digitally generate random variates on a standard Pentium 4 2 GHz computer than generate their DNA-based counterparts.

However, as research continues, scientists are finding more efficient and accurate methodologies by which to synthesize oligonucleotides and systematically read their nucleotide arrays. Thus, it is imaginable the scientists will one day find a methodology by which DNA-based random variates can be generated in the same time constraints as their digital equivalents, but not in the immediately foreseeable future.

In addition to the extended time required to generate and read DNA-based random variates over their digital counterparts, an external transformation is required to produce variates following non-uniform statistical distributions. Since there is no current methodology by which to transform variates in parallel, both digital and DNA-based variates have equivalent conversion times.

Once DNA computing achieves the ability by which to computationally solve complex mathematical equations, DNA-based uniform variates will be capable of being simultaneously translated as opposed to the serial transformation of digital variates. DNA-based random variate generation could then theoretically rival its digital counterparts in speed if the parallel transformation processing negates the additional time requirements to chemically synthesize the oligonucleotide sequences.

8. Conclusion

DNA-based circuit design is continually evolving as DNA paradigms can be developed to represent their digital equivalents. Current efforts of our research team are dedicated to the development of DNA-based methodologies to mimic the digitally based data manipulation counterpart. This work was done in conjunction with a vision that DNA-based circuitry, when the technology matures, will form the basis for a tamper-proof security module, revolutionizing the meaning and concept of tamper-proofing and possibly preventing it altogether based on accurate scientific observations. A paramount part of such a solution would be self-generation of random numbers.

This presentation demonstrates how a microfluidic device can act as a random number generator, demonstrating how data can be generated using DNA sequences. Oligonucleotide synthesis is used to randomly generate a nucleotide sequence, plasmid vectors enable temporary storage, and chromatogram analysis enables the translation from a sequence to its digitally equivalent random number. Long term storage is achieved through spotted microarray fabrication, which enables each sequence's expression levels to be permanently stored.

In order to verify randomness, one must verify that sequences have uniformity and are non-correlated. A wet-lab experiment is required to verify no correlation exists between the previously selected nucleotide and the next randomly selected nucleotide in sequence generation. After generating a multitude of sequences, they must be translated into their digital form through a chromatogram. The sequences can then be evaluated to confirm randomness.

Acknowledgments

C.M. Gearheart would like to thank Hank and Becky Conn for their continued support. This work was supported in part by NIH–NCRR Grant P20RR16481 and NIH–NIEHS Grant P30ES014443. Its contents are solely the responsibility of the authors and do not represent the official views of NCRR, NIEHS, or NIH.

References

Banks, J., Carson II, J.S., Nelson, B.L., Nicol, D.M., 2005. Discrete-Event System Simulation. Pearson Prentice Hall, Upper Saddle River, NJ.
Bogard, C.M., Rouchka, E.C., Arazi, B., 2008. DNA media storage. Progress in Natural Science 18, 603–609.
Fujiwara, A., Matsumoto, K., Chen, W., 2004. Procedures for logic and arithmetic operations with DNA molecules. International Journal of Foundations of Computer Science 15, 461–474.
Hart, S., 1993. Test-tube survival of the molecularly fit. Bioscience 43, 738–741.
Katakai, R., Goodman, M., 1982. Polydepsipeptides. 9. Synthesis of sequential polymers containing some amino acids having polar side chains and (S)-lactic acid. Macromolecules 15, 25–30.
Klug, W.S., Cummings, M.R., 2003. Genetics: A Molecular Approach. Pearson Education, Inc., Upper Saddle River, NJ.
Knuth, D.E., 1997. The Art of Computer Programming, Volume 2: Seminumerical Algorithms. Addison-Wesley Professional.
Lyons, R.H., 2008. Interpretation of Sequencing Chromatograms (accessed 29.02.08) http://seqcore.brcf.med.umich.edu/doc/dnaseq/interpret.html.
Merrifield, R.B., 1963. Solid phase peptide synthesis. I. The synthesis of a tetrapeptide. Journal of the American Chemical Society 85, 2149–2154.
Montgomery, D.C., 2001. Design and Analysis of Experiments. John Wiley and Sons, Inc., New York, NY.
Rukhin, A., et al., 2001. A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications. National Institute of Standards and Technology (Report no. NIST Special Publication 800-22).
Rychlik, W., Spencer, W.J., Rhoads, R.E., 1990. Optimization of the annealing temperature for DNA amplification in vitro. Nucleic Acids Research 18, 6409–6412.
Schneider, T., Hengen, P.N., 2004. Molecular Computing Elements, Gates and Flip-Flops. USA, 37 pp.
Seelig, G., Soloveichik, D., Zhang, D.Y., Winfree, E., 2006. Enzyme-free nucleic acid logic circuits. Science 314, 1585–1588.
U.S. Department of Energy Genome Research Projects, 2008. PRIMER: Genomics and its Impact on Science and Society: The Human Genome Project and Beyond. Oak Ridge National Laboratory.
Weigl, A., Anheier, W., 2003. Hardware comparison of seven random number generators for smart cards. In: ITG-GI-GMM Workshop of Test Methods and Reliability of Circuits and Systems, Timmendorfer Beach, pp. 55–58.