

A SYSTEM FOR CREATING SECRET KEY MATERIAL

James Munro
Digital Lobe, LLC
Rochester, New York

ABSTRACT

Presented is a system for generating secret key bits for encrypting data that are to be sent between electronic devices. In a common scenario, one electronic device is fixed in position (such as a POS terminal), and the other device is hand-held. The hand-held device is swiped past the fixed device, and the profile of the velocity between them is measured simultaneously and independently by the devices. Each device will then have measured identical, and highly unique profiles, which are then digitally processed to obtain a series of secret bits. At no time are data associated with the secret key bits, or their generation, communicated between the two devices. Lastly, the results of computer simulations are presented that show that the secret key bits are random, and can be identically generated by the communicating devices given reasonable SNR's present during the velocity measuring process.

INTRODUCTION

A recurring problem in symmetric cryptography is the distribution of secret keys. Secret keys are required for symmetric encryption and decryption of messages transmitted over an unsecure medium, such as over a wireless radio link or over the Internet. Unfortunately it is problematic to distribute a secret key over a communication channel before that communication channel has been secured.

We propose an alternative mechanism for generation of secret key bits required by electronic communication systems. The secret key bits are independently derived by two or more electronic devices based upon a simultaneous measurement of the physical distance or relative velocity between the devices.

GEOMETRIC SETUP

A common setup in which secret key bits are required is one in which secure communications are needed between two parties of a financial transaction. In this case, one device would be a point of sale (POS) terminal, and the other would be a handheld electronic device in which financial account information (e.g., credit card information) is securely stored. During the transaction the credit card information is transmitted securely, and wirelessly, from the handheld device to the POS terminal.

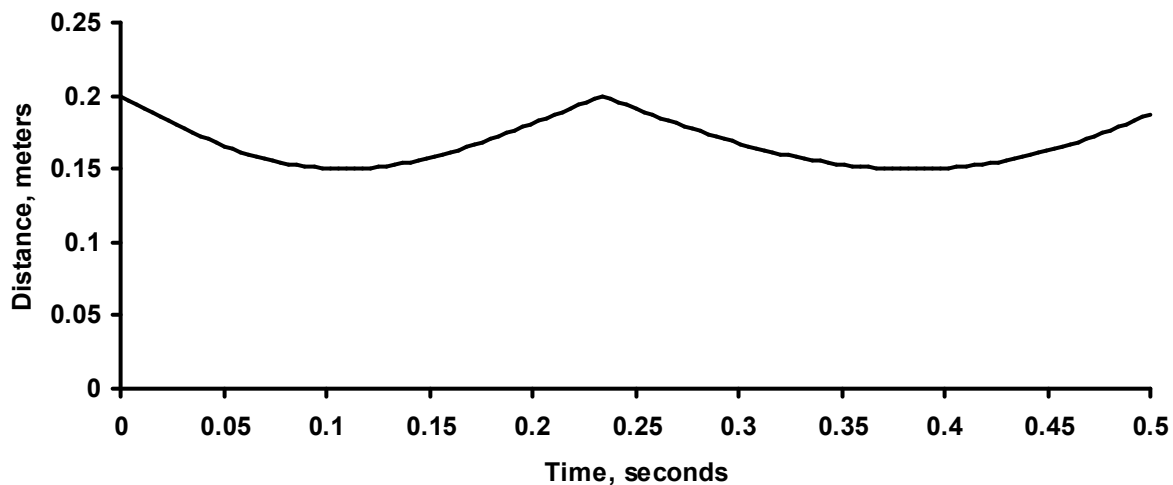
It is straightforward to communicate data wirelessly over an unsecure link, but establishing a secure link for the communication has proven problematic. The main difficulty is that the keys needed for establishing the secure link are derived from public or quasi-public sources, which are available to eavesdroppers who can subsequently decrypt the data being communicated. Therefore it is desirable to use an alternate source from which the secret keys are generated. This source must be known to both the POS and portable devices simultaneously, but not known to the rest of the world. A shared physical

property, such as the distance or velocity between the devices, can serve as the basis for secret key generation. An improvement is to capture the distance or velocity profile over time such that a “swipe” signature is measured that greatly increases the uniqueness of the key source, and also serves as a basis for collecting several key bits. A typical distance profile is shown in Figure 1, below, for a double-pass swipe. For reference, the equation of this graph was arbitrarily chosen to be

$$d = 0.2 - 0.05|(\sin(4\pi t + 0.25\sin(3\pi t)))| \quad (1)$$

and is also used to generate the tabular data presented later for illustration purposes.

Figure 1 Double-Pass Swipe



Path Measurement Process

At the heart of the secret-key generating algorithm is the distance-measuring functional block. The requirements of this subsystem include:

- Low Cost
- Compact, such that it can readily fit into a handheld portable device
- Measure repeatably, such that both devices measure the same distance profile
- Measure quickly, so that several distance measurements can be made during the swipe
- Use emissions that do not carry encryption data

A new distance-measuring technology¹ has recently been introduced that meets all of these requirements. It is based upon the emission of an optical or IR signal that is coherent-burst modulated. This emission is then back-reflected from the face of the opposing electronic device, and then received and processed. The

processing entails computing the distance based upon the phase-shift of the received signal compared to the transmitted signal, according to the formula

$$d = c\Delta\phi/4\pi f_0 \quad (2)$$

where d is the distance, $\Delta\phi$ is the phase shift, c is the speed of light, and f_0 is the burst modulation frequency. There are several methods of computing a phase shift between two signals, but the method that is compatible with the above-mentioned requirements is the phase-computing Discrete Fourier Transform (DFT). This algorithm computes the phase of an arbitrary periodic signal according to the formula :

$$\phi = \tan^{-1} \frac{x_1 - x_3}{x_0 - x_2} \quad (3)$$

where x_0 through x_3 are four sample points collected 90° apart along one period of the received signal. The inverse tangent function can be readily implemented in software on any low cost digital processor, and the data sampling costs can be reduced to irrelevance through the use of equivalent time sampling methods. Furthermore, averaging across several x_n samples or phase measurements can improve measurement repeatability in the presence of noise.

Note that this distance measurement process is based upon measuring the phase difference, $\Delta\phi$, between the transmitted and received signals. Generally it is simple and low-cost to measure one of these signals accurately, but not both as additional optical switching means would be required to first route a portion of the transmitted signal to the processing electronics and then route the received signal to the processing electronics. Therefore, an *absolute* distance measurement between the devices becomes a costlier proposition, whereas a *relative* distance is not. Relative distance measurements, or a change in distance over time, also known as velocity, can be readily calculated by both electronic devices simultaneously, and at low cost. Therefore, swipe velocity data can be used as the basis for the secret-key generation algorithm.

Secret Key Generation

To generate a series of bits that can be used as a secret-key, the swipe profile, or more particularly the velocity of the profile, must be known at a number of locations along the profile. The number of locations is generally greater than or equal to the number of bits needed in the secret-key. For example, if the key is 128 bits in length, then at least 128 velocity points are required.

In the digital processor that computes the swipe profile, the series of velocity points are stored internally as floating point numbers. Note that if the two communicating devices are synchronized at the start of the profile measurement, then the series of velocity points, internal to both devices, will be identical. It is then a simple matter for the processor to convert the floating point number to an integer, and then select one bit from each integer velocity point as a secret key bit. The selection of which bit to use as a key bit is not trivial. Low order bits are heavily influence by noise, and will differ between the electronic devices. On the other hand higher order bits will not be random, but will have serial correlation as well as other statistical defects. It has been found that one can expect between one and three bits, residing near

the center of the integer velocity, to be measurably random, unique in accordance with the swipe profile, yet above the noise floor such that both devices can reliably arrive at the same secret key bits.

In Table I, below, is presented computer-simulated data values of an example of the first several measurements of a swipe profile based upon the swipe distance of equation (1). In this example, 128 key bits are desired, and 129 distance measurement data points are collected. The time between distance measurements is 1/128 seconds. The velocity computation is somewhat different than normal velocity arithmetic, however, in that adjacent distance values were not used. Instead, it has been found that selecting non-adjacent distance values improves the SNR as well as provides an additional degree of randomness to the key bit generation. In the ongoing example, the velocity, more appropriately called a pseudo-velocity or p-velocity, is calculated by using distance values that are 32 points apart, divided by the time between the points, times a scale factor. This formula is:

$$pvel_n = scale \times |(d_n - d_{n+32})| / dt \quad (4)$$

In the ongoing example, scale=1,000,000, 1/dt =128, and equation (4) simplifies to

$$pvel_n = 128,000,000 \times |(d_n - d_{n+32})| \quad (5)$$

The p-velocity value of equation (5) is processed and stored in memory by a digital processor as a floating point number. The fractional bit values of the p-velocity's are subsequently truncated, and the p-velocity's become an integer data type. The bits representing this integer can be sub-divided into three groups as shown in Table II. In this Table, the higher order bits, bits 23 through 11, are not measurably random from p-velocity to p-velocity, and exhibit some degree of correlation or other statistical defects that possibly makes their values predictable due to their position within the p-velocity value. The next groups of bits, bit 10 through 8, are measurably random. The randomness is not due to noise in the measurement process, but instead is due to their position within p-velocity, and vary randomly, and uniquely, with the swipe profile. Note that bits 23 through 8 will be identical for every p-velocity at both electronic devices, but only bits 10 through 8 should be used as key bits because of their random properties. The last group of bits, 7 through 0, will vary randomly due to measurement noise, and will not be identical at the two electronic devices.

TABLE I

Measurement Number	Distance (m)	P-Velocity	Int P-Velocity	Binary P-Velocity
0	0.200000	6230046.88	6230046	010111110001000000011110
1	0.197087	5771826.54	5771826	010110000001001000110010
2	0.194185	5298719.90	5298719	0101000001101101000011111
3	0.191304	4812654.14	4812654	010010010110111101101110
4	0.188454	4315577.52	4315577	010000011101100110111001
5	0.185645	3809448.07	3809448	001110100010000010101000
6	0.182888	3296222.60	3296222	001100100100101111011110
7	0.180192	2777845.89	2777845	001010100110001011110101
8	0.177567	2256240.35	2256240	001000100110110101110000
9	0.175021	1733296.12	1733296	000110100111001010110000
10	0.172563	1210861.64	1210861	000100100111100111101101
11	0.170202	690734.84	690734	000010101000101000101110
12	0.167945	174655.03	174655	000000101010101000111111
13	0.165799	335704.68	335704	000001010001111101011000
14	0.163770	838744.10	838744	000011001100110001011000
15	0.161866	1332942.00	1332942	000101000101011011001110
16	0.160090	1816861.38	1816861	000110111011100100011101
17	0.158449	2289153.94	2289153	001000101110111000000001
18	0.156946	2748563.77	2748563	001010011111000010010011
19	0.155585	3193930.26	3193930	001100001011110001001010
20	0.154369	3624190.26	3624190	001101110100110011111110
21	0.153300	4038379.42	4038379	001111011001111011101011
22	0.152380	4435632.85	4435632	010000111010111010110000
23	0.151611	4815185.05	4815185	010010010111100101010001
24	0.150991	5176369.19	5176369	010011101111110000110001
25	0.150522	5518615.73	5518615	010101000011010100010111
26	0.150203	5841450.44	5841450	010110010010001000101010
27	0.150032	6144491.92	6144491	010111011100000111101011
28	0.150008	6370458.79	6370458	011000010011010010011010
29	0.150129	6076946.41	6076946	010111001011101000010010
30	0.150391	5767551.47	5767551	010110000000000101111111
31	0.150792	5443127.63	5443127	010100110000111000110111
32	0.151328	5104559.38	5104559	010011011110001110101111
33	0.151995	4752758.50	4752758	010010001000010101110110
34	0.152789	4388660.36	4388660	010000101111011100110100
35	0.153705	4013220.42	4013220	001111010011110010100100
36	0.154738	3627410.59	3627410	001101110101100110010010
37	0.155884	3232215.82	3232215	001100010101000111010111
38	0.157136	2828630.64	2828630	001010110010100101010110
39	0.158490	2417655.97	2417655	001001001110001111110111
40	0.159940	2000295.85	2000295	000111101000010110100111
41	0.161480	1577554.53	1577554	000110000001001001010010
42	0.163104	1150433.58	1150433	000100011000110111100001
43	0.164806	719929.15	719929	000010101111110000111001
44	0.166580	287029.56	287029	000001000110000100110101
45	0.168421	147287.12	147287	000000100011111101010111
46	0.170323	582055.16	582055	000010001110000110100111
47	0.172279	1016323.13	1016323	0000111110000001000000011
48	0.174285	1449155.69	1449155	000101100001110011000011

TABLE II

Meas.	Bit	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	0	1	1	1	1	1	1	0	0	0	1	0	0	0	0	0	0	0	1	1	1	1	0
1	0	1	0	1	1	0	0	0	0	0	0	0	1	0	0	1	0	0	0	1	1	0	0	1	0
2	0	1	0	1	0	0	0	0	0	1	1	0	1	1	0	1	0	0	0	0	1	1	1	1	1
3	0	1	0	0	1	0	0	1	0	0	1	1	0	1	1	1	1	0	1	1	0	1	1	1	0
4	0	1	0	0	0	0	0	1	1	1	0	1	1	0	0	0	1	1	0	1	1	1	0	0	1
5	0	0	1	1	1	0	1	0	0	0	1	0	0	0	0	0	0	1	0	1	0	1	0	0	0
6	0	0	1	1	0	0	1	0	0	1	0	0	1	0	0	1	1	1	0	1	1	1	1	1	0
7	0	0	1	0	1	0	1	0	0	1	1	0	0	0	0	1	0	1	1	1	1	0	1	0	1
8	0	0	1	0	0	0	1	0	0	1	1	0	1	1	0	1	0	1	1	1	1	0	0	0	0
9	0	0	0	1	1	0	1	0	0	1	1	1	0	0	1	0	1	0	1	1	1	0	0	0	0
10	0	0	0	1	0	0	1	0	0	1	1	1	1	1	0	0	1	1	1	1	0	1	1	0	1
11	0	0	0	0	1	0	1	0	1	0	0	0	0	1	0	1	0	0	0	1	0	1	1	1	0
12	0	0	0	0	0	0	1	0	1	0	1	0	1	0	1	0	0	0	1	1	1	1	1	1	1
13	0	0	0	0	0	1	0	1	0	0	0	1	1	1	1	1	1	0	1	0	1	1	0	0	0
14	0	0	0	0	1	1	0	0	1	1	1	0	0	1	1	0	0	0	1	0	1	1	0	0	0
15	0	0	0	1	0	1	0	0	0	1	0	1	0	1	0	1	1	0	1	1	0	0	1	1	0
16	0	0	0	1	1	0	1	1	1	1	0	1	1	1	0	0	1	0	0	0	1	1	1	0	1
17	0	0	1	0	0	0	1	0	1	1	1	0	1	1	1	0	0	0	0	0	0	0	0	0	1
18	0	0	1	0	1	0	0	1	1	1	1	1	0	0	1	0	0	1	0	0	1	0	0	1	1
19	0	0	1	1	0	0	0	0	1	0	1	1	1	1	1	0	0	0	1	0	0	1	0	1	0
20	0	0	1	1	0	1	1	1	1	0	1	0	0	1	1	0	0	1	1	1	1	1	1	1	0
21	0	0	1	1	1	0	1	1	0	1	0	0	1	1	1	1	0	1	1	1	0	1	0	1	1
22	0	1	0	0	0	0	1	1	1	1	0	1	0	1	1	1	0	1	0	1	1	0	0	0	0
23	0	1	0	0	1	0	0	1	0	1	1	1	1	1	0	0	1	0	1	0	1	0	0	0	1
24	0	1	0	0	1	1	1	0	1	1	1	1	1	1	1	0	0	0	0	1	1	0	0	0	1
25	0	1	0	1	0	1	0	0	0	0	0	1	1	0	1	0	1	0	0	0	1	0	1	1	1
26	0	1	0	1	1	0	0	1	0	0	1	0	0	0	0	1	0	0	0	1	0	1	0	1	0
27	0	1	0	1	1	1	0	1	1	1	0	0	0	0	0	0	1	1	1	1	0	1	0	1	1
28	0	1	1	0	0	0	0	1	0	0	1	1	0	1	0	0	1	0	0	1	1	0	1	0	0
29	0	1	0	1	1	1	0	0	1	0	1	1	1	1	0	1	0	0	0	0	1	0	0	1	0
30	0	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	1	1	1	1	1
31	0	1	0	1	0	0	1	1	0	0	0	0	0	1	1	1	0	0	0	1	1	0	1	1	1
32	0	1	0	0	1	1	0	1	1	1	1	0	0	0	0	1	1	1	0	1	0	1	1	1	1
33	0	1	0	0	1	0	0	0	1	0	0	0	0	0	1	0	1	0	1	1	1	0	1	1	0
34	0	1	0	0	0	0	1	0	1	1	1	1	0	1	1	1	1	0	0	1	1	0	1	0	0
35	0	0	1	1	1	1	0	1	0	0	1	1	1	1	1	0	0	1	0	1	0	0	1	0	0
36	0	0	1	1	0	1	1	1	1	0	1	0	1	1	0	0	1	1	0	0	1	0	0	1	0
37	0	0	1	1	0	0	0	1	0	1	0	1	0	0	0	0	1	1	1	0	1	0	1	1	1
38	0	0	1	0	1	0	1	1	0	0	1	0	1	0	0	0	1	0	1	0	1	0	1	1	0
39	0	0	1	0	0	1	0	0	1	1	1	0	0	0	0	1	1	1	1	1	1	0	1	1	1
40	0	0	0	1	1	1	1	0	1	0	0	0	0	0	1	0	1	1	0	1	0	0	1	1	1
41	0	0	0	1	1	0	0	0	0	0	0	0	1	0	0	1	0	0	1	0	1	0	0	1	0
42	0	0	0	1	0	0	0	1	1	0	0	0	1	1	0	1	1	1	1	0	0	0	0	0	1
43	0	0	0	0	1	0	1	0	1	1	1	1	1	1	1	0	0	0	0	1	1	1	0	0	1
44	0	0	0	0	0	1	0	0	0	0	1	1	0	0	0	0	1	0	0	1	1	0	1	0	1
45	0	0	0	0	0	0	1	0	0	0	0	1	1	1	1	1	1	0	1	0	1	0	1	1	1
46	0	0	0	0	1	0	0	0	1	1	1	0	0	0	0	0	1	1	0	1	0	0	1	1	1
47	0	0	0	0	1	1	1	1	1	1	0	0	0	0	0	1	0	0	0	0	0	0	0	1	1
48	0	0	0	1	0	1	1	0	0	0	0	1	1	1	1	0	0	1	1	0	0	0	0	1	1

Computer Simulations

A series of computer simulations were conducted to confirm that the dual requirements of simultaneous identical key generation and randomness could be met. In the simulations, 1000 different parabolic (i.e., single pass) swipes were created, and the distance profiles were generated according to the distance measuring method described above. Furthermore, noise was introduced into the distance measuring simulation, such that a signal-to-noise ratio of 500 was realized.

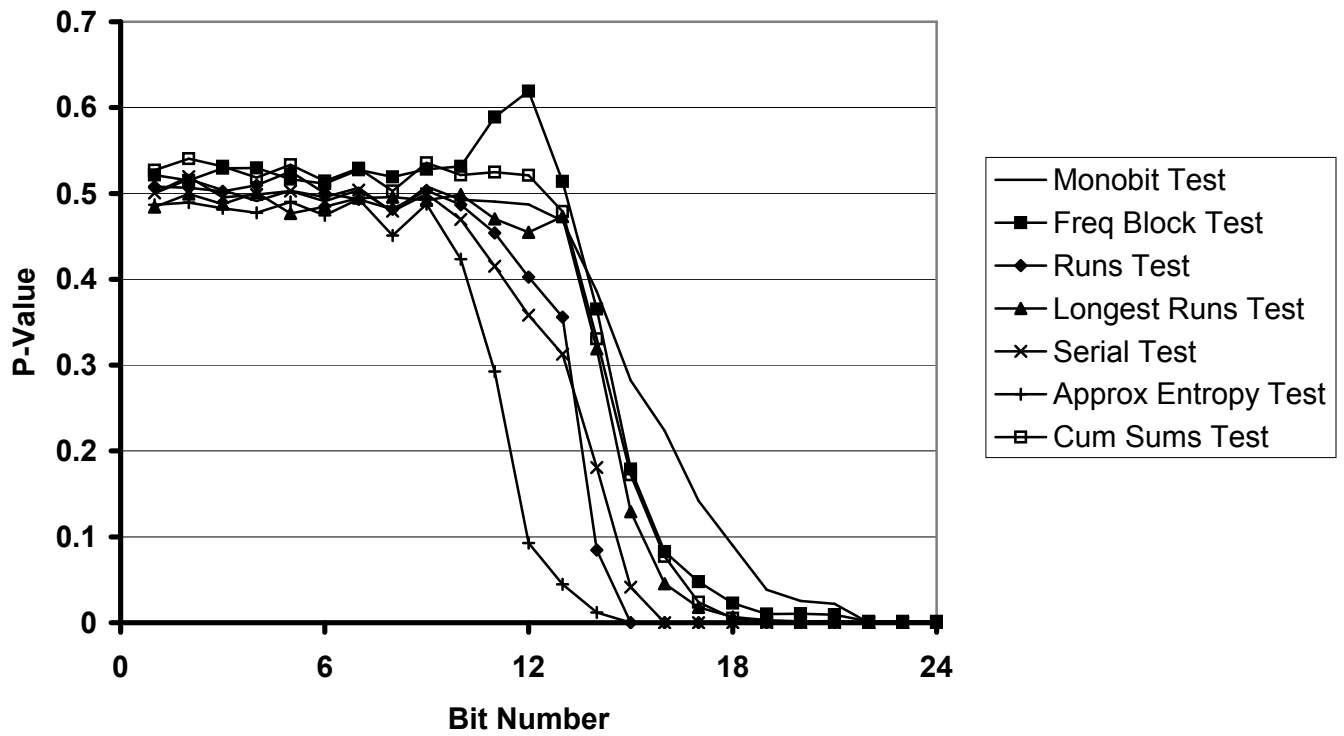
For each individual swipe profile, two simulations were made: one for each device. For each simulation, 129 floating point distance measurements were generated, converted to integer P-velocity data type, and the resulting bits were inspected for randomness and the ability to match the corresponding bit of the parallel simulation.

The tests for randomness were based upon a suite of randomness tests² available from NIST. The tests include:

- **Monobit Test:** A test that determines whether the number of zeroes and ones in a sequence are approximately the same as would be expected for a truly random sequence.
- **Frequency Block Test:** A test that determines whether the frequency of ones in a block of length M is approximately M/2 as would be expected for a truly random sequence.
- **Runs Test:** A test that determines whether the number of runs of ones and zeroes of various lengths is as expected for a random sequence.
- **Longest Runs Test:** A test that determines whether the length of the longest run of ones within a sequence is consistent with the length of the longest run of ones that would be expected in a random sequence.
- **Serial Test:** A test that determines whether the number of occurrences of the 2^m m-bit overlapping patterns is approximately the same as would be expected for a truly random sequence.
- **Approximate Entropy Test:** A test that compares the frequency of overlapping blocks of two consecutive/adjacent lengths against the expected result for a random sequence.
- **Cumulative Sums Test:** A test that determines whether the cumulative sum of the partial sequences occurring in the tested sequence is too large or too small relative to the expected behavior of that cumulative sum for random sequences.

The output of all of these tests is a P-value whose nominal value for randomness is 0.50. Next, 1,000 simulations of 1,000 different swipe profiles were conducted, and the 1,000 P-values were averaged together for each of the seven NIST tests, for each bit of the integer p-velocity. The results are presented in Figure 2, below, in which it is seen that different tests indicate randomness at different bit positions within the integer. The Approximate Entropy Test was the most stringent, with bits 10 and below being measurably random.

Figure II Average P-Values over 1000 Simulated Swipes



Furthermore, for bit position 10 there was one mismatched key sequences between the two electronic devices over the 1,000 simulated swipes, two mismatched keys at bit nine, and five mismatched keys at bit eight. A mismatched key would be interpreted by the user as a “swipe failure” at the POS device, and the swipe would have to be repeated. A failure rate of less than half of a percent would be deemed acceptable in this application, although this failure rate could undoubtedly be improved with anticipated improvements in the key bit generating algorithm.

Summary

A method and algorithm has been proposed for the simultaneous generation of identical secret-key bits at two electronic devices based upon the shared velocity profile between them. The velocity-measuring method and the key-generation algorithm are both simple to implement, and low-cost. At no time is keying material transmitted between the devices. Computer simulations based upon the distance measuring method, using a realistic SNR, has shown that secret key sequences can be generated that are measurably random, and identical at the communicating devices with high probability.

References

¹ Munro, James F, *Low Cost Laser-Rangefinder with Crystal Controlled Accuracy*, Feb 2005 SPIE Journal of Optical Engineering.

² NIST (National Institute of Science and Technology) website <http://csrc.nist.gov/rng/>

About the Author

James F. Munro received a BSEE degree with High Honors from Michigan Technological University in 1982. He also earned an MS, Optics, from the University of Rochester in 1990, and an MBA from the William E. Simon Graduate School of Business, also at the University of Rochester, in 2000. Past positions include optical, electrical, and software engineering, up to and including the title of VP, Engineering. Mr. Munro has professional experience in the fields of phase measuring interferometry, barcode scanners, color measuring instrumentation and spectrophotometers, and microstructured optical films. Mr. Munro is also the founder and CTO of Munro Ranging, LLC, and a cofounder of Digital Lobe, both located in the Rochester, NY area.