Cyber-Secure UAV Communications using Heuristically Inferred Stochastic Grammars and Hard Real-Time Adaptive Waveform Synthesis and Evolution

Stuart H. Rubin[1], William K. Grefe[1], Thouraya Bouabana-Tebibel[2], Shu-Ching Chen[3], Mei-Ling Shyu[4], and Kenneth S. Simonsen[1]

[1]Space and Naval Warfare Systems Center Pacific, San Diego, CA 92152-5001, USA
[2]Ecole Nationale Supérieure d'Informatique, LCSI Laboratory, ALGERIA
[3]Florida International University, School of Computing and Information Sciences, Miami, FL 33199, USA
[4]University of Miami, Department of Electrical and Computer Engineering, Coral Gables, FL 33124, USA
[1]{stuart.rubin, william.grefe, kenneth.simonsen}@navy.mil, [2]t_tebibel@esi.dz, [3]chens@cs.fiu.edu, [4]shyu@miami.edu

Abstract— **This paper initially describes how an inferred context-free (stochastic) grammar can be used to verify command transmissions and serve as a hedge against a successful cyber-attack. The remainder of the paper addresses a computational problem not amenable to closed-form solution; namely, the hard real-time (~57 usec) synthesis of a desired waveform through the adaptive modification of a carrier wave. This effectively increases the signal to noise ratio – ensuring better UAV communications. Here, the modulation of the primary waveform is under user control and is of strictly positive amplitude. The primary waveform induces a secondary waveform having delayed leading and trailing edges and expanded rise and fall times. There is, in general, a direct relation between the period of the primary waveform and the amplitude of the secondary waveform. The relation between the primary and secondary waveforms may be characterized by trigonometric functions or even interpolating polynomials. However, response time will be minimized where the primary waveforms are discretized and stored in the form of array-based cases. The tertiary (target) wave may be any periodic trigonometric function, but is taken to be a simple sine wave without loss of generality. The task of the adaptive program is to minimize ||*s(t)* – *g(t)*||2, where *f(t)* → *g(t)* and *f(t)* is the primary waveform at time *t*, *g(t)* is the secondary waveform at time *t*, and *s(t)* is the tertiary waveform at time *t*. A computationally efficient algorithm is provided for solving this task in real time. Moreover, an evolutionary program (EP) is provided for automatic case acquisition. Primary waveforms are mutated in accordance with a normal distribution.**

Keywords – Adaptive Programming, Case-Based Reasoning, Evolutionary Programming, Expectation-Driven Receiver, Grammatical Inference, Hard Real-Time Systems, Waveform Analysis and Synthesis

I. INTRODUCTION

The goal of this paper is define a technique to cyber-secure UAVs and UAV swarms, which are subject to jamming and hijacking, in some instances. Methodological descriptions will follow. One advantage, provided by autonomy, is that the known commands sent to and from UAVs are well-defined. In fact, although one cannot know a priori exactly what command was intended, one can create a grammar, or qualitative technique, for delimiting the space of communication sequences. The idea is that the subsequences that are received can be used to delimit the interpretation of those, which are garbled. While a Monte Carlo analysis would be of use here, the autonomous design allows for the use of a stronger predictor – the context-free grammar as applied to sentential forms.

For example, the command, "drone 1 go left, drone 2 go right, drone 3 go up, and drone 4 go ????" could be a sentential form of the grammar if and only if drone 4 was ordered to go down. Otherwise, the grammar would specify, "drone i *and* 4 go {left, right, up}", i ≠ 4. The grammar can reduce the ambiguity in the interpretation by often reducing the number of possible interpretations of a distorted transmission. Transmissions are not uniformly distorted. Thus, the packets, which are received can be used to associatively reduce the ambiguity – like fitting one piece in a puzzle enables a chain of pieces to be fit. A stochastic grammar can also be induced [1]. Here, there may be more than one sentential alternative ; and, each can be ascribed a probability. The product of the probabilities is then interpreted to imply the chance of an erroneous reception.

Context-free grammars can be inductively inferred through the use of heuristics [2] [3]. They are not going to provide stand-alone cybersecurity. Rather, they are part of a mix of tools, which collectively provides cybersecurity. Semantic randomization is another such tool [4] – [6].

A related quantitative technique can be devised to minimize ambient noise and overcome relatively low-power jamming. The qualitative and quantitative techniques are additive. Taken together, these techniques vie against the hijacking of a UAV(s).

The quantitative adaptive technique involves the rapid real-time minimization of $\| s(t) - g(t) \|_2$, where $f(t) \rightarrow g(t)$ and *f(t)* is the primary waveform at time *t*, *g(t)* is the secondary waveform at time *t*, and *s(t)* is the tertiary waveform at time *t* [7]. The inverse waveform, $g^{-1}(t)$ gives $f(t)$ and is computed using a table lookup – so it always

exists. The use of a table lookup will serve to enhance the performance of the system. Different memory management schemes may be used to randomize the representation of the primary waveform, but all have an associated de-convolution penalty – making them less of an option. One possibility is to make use of Denning's cache concept whereby the most frequently used (MFU) primary waveforms are retrieved in their de-convolved form. Less frequently used (LFU) waveforms are stored in a hierarchy of convolved forms and are de-convolved (one level) upon reference. The basis for "aging" the cache is an assigned penalty function for de-convolution. That penalty function is determined by the cost or metric for the inconvenience of having asynchronous discontinuities (spikes) in the synthesized continuous target function (i.e., tertiary waveform). The use of sparse matrix techniques for convolving the primary waveform, while providing for much better utilization of the memory space, will slow the system down in view of the need for additional address calculations (de-convolution). The use of numerical quadratures such as trigonometric functions or collocating polynomials would be similar.

II. EXPECTATION-DRIVEN RECEIVERS

This section will provide a brief illustration of the use of an inferred stochastic grammar. Again, the purpose served by such a memory is to delimit garbled text by providing a context for its likely interpretation. A stochastic likelihood may also be provided. Here, any likelihood below a set squelch is dismissed as garbled. However, this does not insure that all above squelch are valid. For this reason, we do not advocate stochastic verification for most domains of interest.

To begin, it may be assumed that linguistic communications do not fit a static template, but rather evolve – just as our use of English does to a greater or lesser extent. The evolution of any grammar of higher order than regular (e.g., context-free) requires a heuristic approach [2] [3]. Consider the following sentential forms, which would be obtained from recorded conversations between the user and the drone(s) as well as drone to drone conversations.

1. Drone 1 go left.
2. Drone 2 go right.
3. Drone 3 go up.
4. Drone 4 go down.
5. Drone 1 and 4 go left.
6. Drone 1, 2, and 4 go right.
7. Drone 2 and 3 go up.

These sentential forms may be randomized [4] to produce the following sets, where set mnemonics may or may not be properly named, depending on the implementation.

- Number = {1, 2, 3, 4}
- Direction = go {left, right, up, down}.

A context-free grammar randomizes the sentential forms through the use of the randomized sets. Note that the optimal allocation of sets and elements is generally not deterministic and involves search, which is rendered tractable, in general, through the use of heuristics. Here is such a grammar, which is not necessarily unique:

S → Drone A
A → Number Direction | Number and Number Direction | Number, Number, B
B → Number, B | and Number Direction

Next, every received instruction can be tested for conformity with this grammar. This adds credence, but not certainty that what is received is what was sent. Although still subject to error, this error is generally far less than that associated with the use of parity bits. For example, the following sentential forms would be deemed to be erroneous.

1. Go left Drone 2.
2. Drone 1 go right 3 go left.
3. Drone 3, and 4 go up.

Notice however that there is nothing to preclude repetition of numbers as in, "Drone 1 and 1 go left." This can be handled by a higher-order, or context-sensitive grammar (e.g., the classic $a^n b^n c^n$ problem), but this adds an unnecessary level of complexity. It is unnecessary because it requires more computational time, stronger heuristics, and does little to increase cybersecurity as a tradeoff. Rather, it is recommended not to be included; and, if it need be included, use production rules – e.g., as follows.

- $\forall Number\ (i, j) \neg (i, i)$

The use of stochastic grammars would involve assigning each context-free production (i.e., including each or'd production) a probability of being fired. Then, a sentential form found to be in the grammar would have the product of the applied productions as its probability. The more common the production, the higher its ascribed probability (e.g., using a simple machine learning algorithm to find for this). Then, the computed probability represents the likelihood of a sentential form being the one that was transmitted. Again, we don't recommend the use of stochastics here because it vies against the reception of valid, albeit infrequently transmitted, sentential forms. However, there can be application-specific domains, where this is useful because the validity squelch is known to vary with the situational context and because anything less likely can be attributed to a cyber-attack.

One more thing is to consider the possibilities for error correction (e.g., analogous to Gray codes, which may be used for bit corrections). Here again, while it is possible to produce a meaningful error message, it is eminently more practical to ask for a retransmission. For example, “Go left Drone 2” would use the S production to produce the error message, “Missing Drone Number”. The erroneous sentential form, “Drone 1 go right 3 go left” would use the A production to produce the error message, “Unexpected End of Command”. The erroneous sentential form, “Drone 3, and 4 go up” would use the A production to produce the error message, “Missing Drone Number”. These errors provide likely evidence of an attempt to hijack a drone. Even though transmission may not be possible, this information is valuable for it can trigger an organic mission recovery program, which may bring the UAV(s) back into secure communication range (e.g., through the use of GPS in a non-denied environment).

III. SYNTHESIZED WAVEFORMS

Figures 1 thru 4 depict an exemplary discretized grid showing the primary waveform (F) in red and the secondary (lower) waveform (G) in blue. Figure 5 shows the tertiary waveform (S) in green and the secondary waveform again in blue. Note that only primary waveforms need be stored in primary memory – allowing for the depictions. The mesh size is governed by the memory requirements as well as the tolerance for error. That is, the less the tolerance for error, the finer the mesh size and the concomitantly greater the memory requirements; while, the runtime remains practically invariant due to the use of random addressing on the memory bus. Of course, the use of a finer mesh size implies that a greater number of array references will be needed to retrieve a primary waveform of the same duration. This problem can be mitigated to some extent through the use of anticipatory caching; whereby, a limited number of primary waveforms are pre-fetched and cached for a coarser mesh size in anticipation of being referenced as discussed above. The best mesh size is hardware dependent and thus must be chosen experimentally.

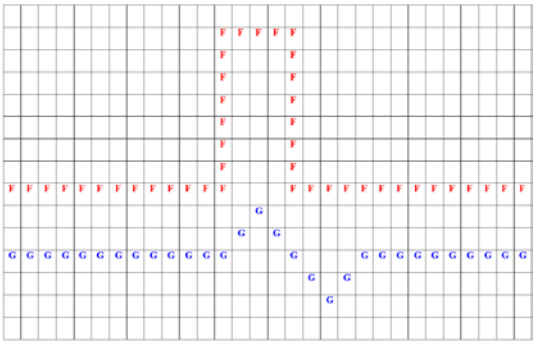

Figure 1. A Dirac Impulse Primary Waveform

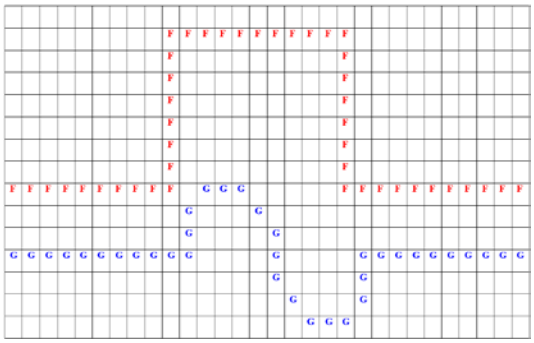

Figure 2. Effect of Increasing the Primary Pulse Width

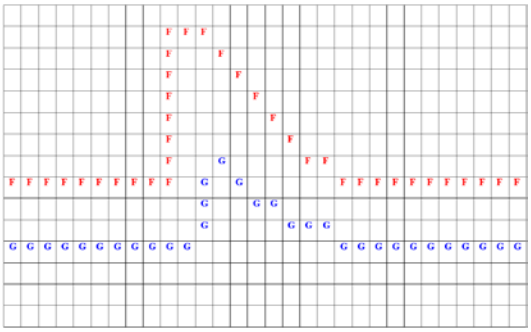

Figure 3. Effect of Varying Primary Edge Fall Rate

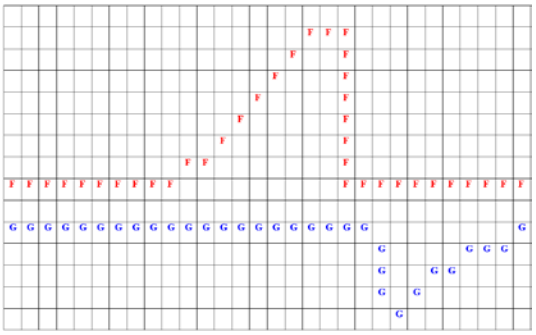

Figure 4. Effect of Varying Primary Edge Rise Rate

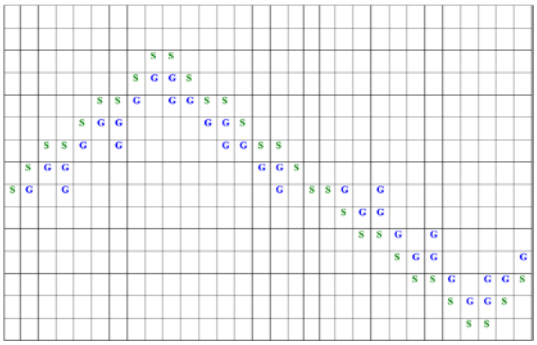

Figure 5. A Tertiary Waveform Approximated by a Secondary One

IV. A HARD REAL-TIME ADAPTIVE ALGORITHM

In this section, we present an adaptive algorithm for generating the tertiary waveform using the secondary waveform as impulsed by the primary waveform. Again, the algorithm was designed to operate under hard real-time constraints. It is tacitly assumed that the effects of sequential secondary pulses are context free; although, using a *k*-limited table-driven approach, the effects of say all pairs of sequential secondary waveforms (e.g., *k*=2) may be similarly stored and indexed (see below); albeit, at exponential space requirements that are $O(k!)$. Thus, delimited *k*=2 "look-back" serves as a practical maximum contextual value.

It should be noted that this algorithm allows for the ready maintenance and extension of the primary-secondary waveform pairs as a consequence of the case-based methodology used. It follows that the case pairings can be automatically acquired through a random exploration and feedback loop (Figure 6).

There is a need to uniformly explore the space of primary waveforms. That is, one must set a squelch where, $g \mid \forall g \in G, \| s(t) - g \|_2 > \delta$ and $g' \mid \| s(t) - g' \|_2 \leq \delta$, where δ is a suitably chosen threshold constant. If the two-norm is less than or equal to this constant, then we say that the secondary impulse response is already covered by the case base. Otherwise, the primary-secondary waveform pairs are saved in the case base. If this constant is chosen to be too close to zero, then the case base will grow to be too large and either not cover the desired primary waveform space, or require excessive memory and/or a coarser mesh size – all of which would be undesirable. Conversely, if this constant is chosen to be too large, then the case base may still cover the desired primary waveform space, but it will necessarily be limited in the number of basis waveforms that it can acquire and, as a consequence, tend to fit the tertiary function with spikes above the desired function, followed by spikes below it (i.e., great volatility), which is deemed to be undesirable. Again, the threshold constant, δ, needs to be chosen experimentally.

Next, a unified algorithm is presented, which will (a) synthesize a target tertiary waveform, (b) use one or more co-processors to effect Gaussian mutations of the best primary waveform to evolve a new one and save it in the case base whenever the value of δ is exceeded by all of the existing case base, and (c) interlace the synthesis of the target tertiary waveform with the synchronized test of a secondary waveform impulsed by a mutated primary waveform. The periodic variance in the synthetic tertiary waveform can be automatically ignored by a properly designed receiver, which can be synchronized by a train of reference pulses. The co-processor(s) are allowed to run until g' is found, or they are preempted by the arrival of a new problem. The justification here is that successful discovery is some decreasing exponential function of time in as much as the mutation function is not uniformly distributed (i.e., it is normally distributed). Thus, there is a better chance of discovery, given limited resources, if one pages out the old problem and pages in the new. The old problem may be cached and subsequently paged in (subject to the limitations imposed by thrashing) whenever a processor resource is freed. A time-stamp mechanism is used to expunge the oldest problems where called for by limitations in storage space. The approach taken here is thus seen to be unique in that it interlaces a hard real-time adaptive wave generator with an evolutionary case-based acquisition system, which learns to better model the tertiary waveform over time! The essential pseudo-code for this algorithm follows.

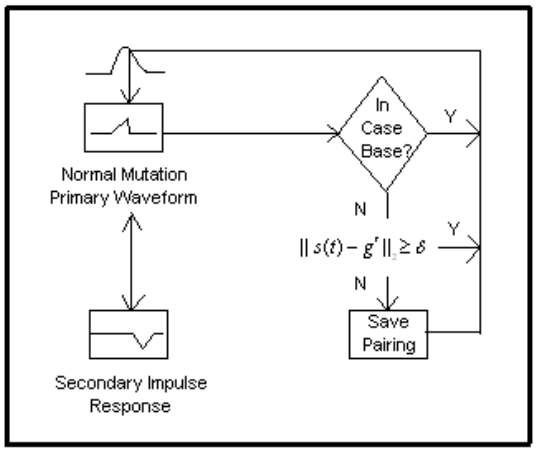

Figure 6. Conceptual Evolutionary Acquisition of Primary-Secondary Waveform Pairings

1. Start:
2. Label all secondary waveforms with the amplitude of their spike.
3. Store these amplitudes in an array such that each amplitude is paired with its defining secondary waveform and thus with a primary waveform as well. The amplitudes are stored as intervals; so for example, instead of say amplitudes of 1, 2, 3, … one would store these amplitudes as [0.5, 1.5), [1.5, 2.5), [2.5, 3.5), and so on to facilitate pattern matching. (The behavior approaches continuity as

the number of primary functions tends towards a very large number.)

4. L1: Compute the projected amplitude needed at time, *t+1* to approximate the known tertiary waveform (i.e., using the tail of the preceding secondary one for a starting point). Reference the array with this value to retrieve the index of the best primary waveform.
5. Generate the primary waveform.
6. Apply the associated secondary waveform and measure its effect.
7. If $\| s(t) - g \|_2 \geq \delta$, when the next synchronized reference pulse wakes up the co-processor, generate g' using a normally distributed mutation (see section III).
8. Else If the cache is empty, then goto L2
9. Else when the next synchronized reference pulse wakes up the co-processor, retrieve the youngest problem from cache and generate g' using a normally distributed mutation.
10. If $g' | \| s(t) - g' \|_2 \geq \delta$, then time-stamp and cache the problem
11. Else save the solution in the case base
12. If the cache is full, implement a Least-Recently-Used (LRU) memory management policy, which is based on the time stamps.
13. L2: Poll the interrupt vector.
14. Goto L1.
15. End.

V. THE MUTATION ALGORITHM

In its simplest form, this algorithm operates on the assumption that the state space wherein the solution lies has few local maxima. In the search for the single global maxima, the algorithm makes an implicit assumption that climbing the hill that it is currently on will lead to the desired result. This occurs by initializing the primary waveform to a random configuration and thereafter continuously varying this waveform based on a normal distribution around the current definition. The distribution is used to determine such parameters as pulse width, amplitude, and rates of rise and fall on the leading and trailing edges. Exploration in this manner takes place in rounds. The pseudo-code for this algorithm follows.

1. Start:
2. Initialize the best primary waveform w_{best} to one having minimal $\| s(t) - g \|_2$, or to a random continuous waveform if the case-base is empty. Initialize the normal distribution d_i to the flattest distribution available, d_0. Set *found-better* to False.
3. Repeat *i* times.
    a. Vary each defining coordinate in the primary waveform according to d_i, creating primary waveform w_{temp}.
    b. Test this primary waveform according to the evaluation formula, $f(g') = \| s(t) - g' \|_2$
    c. If $f(w_{temp}) < f(w_{best})$
        i. Set $w_{best} \leftarrow w_{temp}$
        ii. Set *found-better* to True.
    d. Goto step 3a.
4. If (*found-better* = True)

a. $d_i \leftarrow d_{i+1}$ (decrease the st. dev of the normal)

b. Set *found-better* to False

c. Goto step 3.

Else

If ($d_i = d_0$)

a. Return w_{best}

b. Exit

Else

a. $d_i \leftarrow d_{i-1}$ (increase the st. dev of the normal)

b. Set *found-better* to False

c. Goto step 3.

5. End.

A. Simulation Results

Three trails were conducted as follows [8]. Trial 1: a random primary waveform is tested against 100 strategies – (10, 100 and 1000) waveforms are tested during a cycle. The new waveforms generated during a cycle are based on the waveform of the best performer against the 100 strategies found. These waveforms are varied from the best performer according to a normal distribution. If during the course of a cycle a primary waveform out-performs the best performer, it is declared the best performer, and the normal distribution is set to be tightened on the next cycle. If however, no primary waveform outperforms the best performer, after an entire cycle, the normal distribution is then relaxed on the following cycle. The normal distribution is tightened by decreasing the standard deviation, and loosened by increasing the standard deviation of the normal. The normal distribution is initially set to the loosest (most like a uniform distribution) standard deviation. There are 8 different normal distributions utilized in Trial 1. Their standard deviations are as follows: [92, 64, 45, 32, 23, 16, 11, 8]. If a better weight set is found while utilizing the tightest distribution, that distribution is maintained for the following cycle. The trial terminates when no new best performer is discovered using the most relaxed normal distribution.

Trial 2: Tests are conducted in a manner similar to Trial 1. After a new best performer is discovered, the normal distribution is set to be tightened to the highest level possible (i.e., standard deviation of 8). In Trial 1, distributions are increased by only a single level after such an event.

Trial 3: Tests are conducted in a manner similar to Trial 1. There are only 4 different normal distributions utilized in Trial 3. Their standard deviations are as follows [96, 32, 16, 8].

B. Tabular Results

The results of these trials have been published in [8]. They are reproduced in Table 1.

Table 1. Results from Testing Different Parameters of the Algorithm

	Sets of 10	Sets of 100	Sets of 1000
Trial 1	87	83	148
	25	82	94
	149	47	119
	32	86	
	-68	138	
	Average 56.2	*Average 87.2*	***Average 120.3***
Trial 2	113	99	164
	65	143	138
	47	113	179
	30	115	
	62	113	
	Average 63.4	*Average 116.6*	***Average 160.3***
Trial 3	110	175	91
	29	117	173
	42	132	
	128	127	
	-88	147	
	Average 44.2	*Average 139.6*	*Average 132*

C. Cybersecurity

Compromised codes can be found by testing a function for equality with the source version stored in an immutable repository. Code executions can be tested and periodically compared with immutable executions. The number of such executions is proportionate to the amount of security that can be had. The system can repair itself if a cyber-attack occurs. The repository comprises firmware, which holds the original unadaultered functions. The system can be restarted to an earlier point and the involved functions restored to validity. The repository may only be written to with the permission of a capability access model.

VI. Conclusion

Signals are filtered through the use of an inferred stochastic grammar. Noise and jamming signals are effectively removed through the use of an expectation-driven receiver. Cybersecurity also entails maximizing the signal to noise ratio. Evolutionary methods have generally not been compatible with hard real-time systems. This paper also presents a new case-based approach that combines hard real-time adaptive algorithms with evolutionary programs capable of modifying and/or augmenting the case base in real time through the use of one or more co-processors (for maximal efficiency). These techniques are expected to find application in adaptive optics, correcting for frequency drift in cellular transmissions, and in boosting signal gain through the improved cancellation of ambient noise or cyber jamming. Applications such as these will serve to enhance military network operations (e.g., ForceNet).

ACKNOWLEDGEMENTS

Stuart would like to acknowledge the assistance of Professor Thouraya Tebibel, Professor Shu-Ching Chen, Professor Mei-Ling Shyu, and our Chief Engineer, William Grefe. Stuart would also like to extend his thanks to the Office of Naval Research (ONR) for providing financial backing for this research with the support of Ken Simonsen. He also extends thanks to division head Ayax Ramirez and branch heads Jamie Lukos and Joanna Ptasinski. Finally, a special thanks to patent attorney Eric J. Anderson. This work was produced by U.S. government employees as part of their official duties and is not subject to copyright. It is approved for public release with an unlimited distribution.

REFERENCES

[1] K.S. Fu, *Syntactic Pattern Recognition and Applications*. Englewood Cliffs, NJ: Prentice-Hall Advances in Computing Science and Technology Series, 1982.

[2] R. Solomonoff, A New Method for Discovering the Grammars of Phrase Structure Languages, *Proc. Int. Conf. Information Processing*, UNESCO Publishing House, Paris, France, pp. 285-290, 1959.

[3] R. Solomonoff, A Formal Theory of Inductive Inference, *Inform. Contr.*, pp. 7 1-22 and 224-254, 1964.

[4] S.H. Rubin and T.-B. Tebibel, "NNCS: Randomization and informed search for novel naval cyber strategies," in *Recent Advances in Computational Intelligence in Defense and Security*, Studies in Computational Intelligence, R. Abielmona *et al*. (eds.) vol. 621, Springer International Publishing, Switzerland, pp. 193-223, 2016.

[5] S.H. Rubin, Cyber-Secure Bootstrapped Two-Level Natural Language Mediated Knowledge-Based Systems of Systems for the Spiral Development of Autonomous Software, NC No. 104882, recd. 24 Oct. 2016.

[6] S.H. Rubin, Cyber-Secure Natural Language Based Learning Intelligent System Shells, NC No. 104782, recd. 12 Oct. 2016.

[7] S.H. Rubin, *Hard Real-Time Adaptive Waveform Synthesis and Evolution*, U.S. Patent No. 7,623,410, 24 Nov. 2009.

[8] S.H. Rubin, *Randomization for Cyber Defense*, Patent Disclosure, NC 103845, 28 Oct. 2015.