Fusing Natural Computational Paradigms for Cryptography. Or, How to Create Quantum Solvable Cryptographic Problems with Heuristic Search

John A. Clark, Susan Stepney
Dept. of Computer Science, University of York, Heslington, York, YO10 5DD, UK
Email: [jac,jeremy,susan]@cs.york.ac.uk

***Abstract*— Recent years have seen the application of evolutionary and other nature-inspired search approaches to achieve human-competitive results in cryptography. We have also seen the emergence of quantum computation as a tremendously exciting computational paradigm with significant potential applications to cryptography. To date there seems to have been no synergistic application of these techniques to cryptographic problems. All applications are geared to the effective exploitation of one computational paradigm or another. Nature-inspired search and quantum computing can, however, be combined to achieve results neither is capable of individually. All that is needed is that classical search get 'close enough' for quantum search to take over and solve the residual problem. This observation has significant implications for the security of crypto-systems and our understanding of the real power of nature-inspired search.**

I. Introduction: Nature-Inspired Computing and Computing by Nature

Nature-inspired search techniques such as genetic algorithms [2] and simulated annealing [5] have shown their worth over a huge number of engineering disciplines. It comes as no surprise that they have been investigated as cryptanalysis tools. Most work has been concerned with elementary ciphers [12], [4], [7], [10] but recent work has included attacks on modern day systems. Pointcheval [8] gives results of attacks using simulated annealing on his PPP-based zero-knowledge schemes. Knudsen and Meier [6] have improved on those results using an unusual and sophisticated attack. Clark and Jacob applied 'problem warping' and a 'timing attack' (essentially using the information obtained by analysis of the trajectory of a simulated annealing search) to the PPP problem. Recently work by Hernandez et al has sought weakness in the TEA block cipher. Gradually, nature-inspired search is being taken into the heart of modern day cryptology.

Quantum computation is one of the most exciting developments of recent years. Here fundamental laws of physics (quantum mechanics)are exploited to produce one of the most powerful computation developments in recent years. Peter Shor's polynomial time algorithm for factorisation is widely regarded as the 'killer application' but other significant results have cryptological application. Grover's search algorithm [3] gives a quadratic speed up over enumerative search. Thus a state space of size $O(2^N)$ can be searched in $O(2^{\frac{N}{2}})$ operations. For many currently existing key sizes (e.g. 80 bits or less), this will render attacks on block cipher key spaces feasible. However, larger secrets will not yield to feasible application of Grover's search.

There are growing bodies of work in applying nature-inspired search and quantum computation to cryptological problems. However, there is no work that we know of that demonstrates that the two can be used *together* to achieve results unattainable by either approach alone. We shall argue that nature-inspired search can be used to **reduce** problems to enable 'the rest' of the problem to be attacked with quantum search approaches such as Grover's search. Thus, the aim of applying a nature-inspired approach is to *create* problems: create problems that we can solve by other means.

We shall illustrate our point by reference to attacking a proposed zero-knowledge scheme by David Pointcheval: the Permuted Perceptron Problem.

II. Preliminaries

A. The Perceptron Problem and Permuted Perceptron Problem

In 1995 Pointcheval [8] suggested what seems a promising scheme based on the *Perceptron Problem* (PP). In fact, he chose a variant of this problem that is much harder to solve known as the *Permuted Perceptron Problem* (PPP). If instances of these problems can be solved the identification schemes are broken. The protocols used to implement the identification schemes are not described here (the reader is referred to [8] for details). This paper concentrates on attacking the underlying NP-complete problems. The notation of [8] and [6] will be used. A column vector whose entries have value +1 or -1 is termed an ϵ -vector. Similarly, a matrix whose entries have value +1 or -1 is termed an ϵ -matrix.

- **Perceptron Problem:** :
  Input: An m by n ϵ-matrix A.
  Problem: Find an ϵ-vector V of size n such that $(AV)_i \geq 0$ for all $i = 1, ..., m$.

- **Permuted Perceptron Problem:**
  Input: An m by n ϵ-matrix A and a multiset S of non-negative numbers of size m.
  Problem: Find an ϵ-vector V of size n such that $\{\{(AV)_i | i = \{1, ..., m\}\}\} = S$.

In the PP we require that image elements $(AV)_i$ be non-negative, in the PPP we require that these elements

have a particular distribution (histogram). If n is odd (even) then the $(AV)_i$ must all take odd (even) numbered values. Pointcheval's PPP schemes used only odd values for n (see below). It is always possible to generate feasible instances of these problems. The matrix A and column vector V are generated randomly. If $(AV)_i < 0$ then the elements a_{ij} of the ith row are negated. This method of generation introduces significant structure into the problem. In particular, the majority vectors of the entries for columns of A are correlated with the corresponding elements of V (as indicated in [8] and [6]). The security of the scheme relies on the computational intractability of exploiting this structure.

Any PPP solution is obviously a solution to the corresponding PP since the PPP simply imposes an extra histogram (multiset) constraint. A solution to the PP is not necessarily a solution to a related PPP. Pointcheval investigated the complexity of generating PPP solutions by the repeated generation of PP solutions. He indicated that matrices of the form $(m, n) = (m, m + 16)$ gave best practical security and offered three particular sizes: $(101, 117)$, $(131, 147)$ and $(151, 167)$.

III. Applying Quantum Search Directly

A. Grover's Search Algorithm

Grover's algorithm searches an unstructured database of size to find a value x that satisfies some predicate GP(x). Thus we could search over a space of keys to find a key that encrypted a given plaintext as a given ciphertext.

We shall make use of the quantum gate U_P which tests the truth of predicate P. Thus we have

$$U_p : |x, 0\rangle \rightarrow |x, P(x)\rangle \tag{1}$$

Quantum registers can be placed in a superposition of possible states. Thus we can easily set up the following superposition:

$$\frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} |x, 0\rangle \tag{2}$$

Here the possible states of the system comprise a concatenation of the n-bit value x and the single bit value 0. The superposition of these states is uniform: when we *observe* the system it will collapse to each of these states with probability $\frac{1}{2^n}$ (the square of the magnitude of the complex amplitude of that state).

Quantum computing allows quantum operations to be applied in one step to all states in a superposition. This quantum parallelism is the source of great power. Here we can apply U_p to the whole superposition:

$$U_P(\frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} |x, 0\rangle) = \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} |x, P(x)\rangle \tag{3}$$

This gives a uniform superposition of states of the form $|x, P(x)\rangle$. The current amplitude of each state in the superposition is still $\sqrt{\frac{1}{2^n}}$ and so the probability of observing each state is $\frac{1}{2^n}$. We wish to find a state with $P(x) = 1$, i.e. one that satisfies the property of interest. The trick is to apply quantum operations to the superposition so as to greatly increase the magnitude of the amplitudes of desired states $|x, 1\rangle$ and greatly decrease those for the states $|x, 0\rangle$. We can then measure the final qubit. If a 0 is observed we must repeat the algorithm again. If a 1 is observed then the state collapses to a superposition of states with final value 1. Further observations on the n qubits encoding x will reveal an x that satisfies the predicate P(x).

Grovers algorithm can be described below (see Rieffel and Polak's excellent introduction to quantum computing for details [9]):

1) Prepare a register in a superposition of values $\frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} |x, 0)\rangle$
2) Compute $P(x)$ on the register as above to give $\frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} |x, P(x))\rangle$
3) Change any complex amplitude a_x to $-a_x$ for any x such that $P(x) = 1$. (Efficient algorithms exist to do this.)
4) Apply inversion about the average to increase the amplitudes of x with $P(x) = 1$. (Efficient algorithms exist to do this.)
5) Repeat steps 2 through 4 $\frac{\pi}{4}\sqrt{2^n}$ times
6) Read the result.

Thus Grover's algorithm returns a solutions $O(2^n)$ operations. Assuming only a single solution satisfying P(x) exists we *fail* to observe that value x with probability 2^{-n}. If more solutions exist then variants of the algorithm can be used that return one of the states.

B. Can Quantum Search Solve this PPP Directly?

Grover's algorithm can be applied to the state space to give quadratic reduction. For problems of size $(101, 117)$ Grover's search should find a solution in $O(2^{\frac{117}{2}})$. Similarly for sizes $(131, 147)$ and $151, 167)$ Grover's will find an answer in $O(2^{\frac{137}{2}})$ and $O(2^{\frac{167}{2}})$ operations respectively.

Although this provides a very significant reduction the computational requirements are still huge. Also, larger sizes of the problem could be adopted to place security beyond computational capability. Other schemes such as Syndrome Decoding [11] might have secret spaces of 512 bits or more, making a direct attack utterly infeasible. One is left with the impression that for problems such as these a direct attack by quantum search is not the way to proceed. Exploiting problem structure by quantum search is still very much in its infancy (Tadd Hogg has provided some results on SAT problems) and there is no clear way to do this for this problem.

Let us proceed to use nature-inspired search, to see if it fares better.

IV. Simulated Annealing

Simulated annealing is a combinatorial optimisation technique based loosely on the physical annealing process of molten metals. An informal description is given below followed by a detailed one.

States and State Cost. Candidate solutions to the problem at hand form the states over which the search will range. With each state V is associated a cost, $cost(V)$, that gives some measure of how undesirable that state is (in physical annealing a high energy state is undesirable). In attacking the Perceptron and Permuted Perceptron Probems, the current state (solution) will be some ϵ-vector V_{curr} of size n. The choice of cost function is a crucial issue (discussed in Section **??**).

The Neighborhood. The search moves from state to state in an attempt to find a state V_{best} with minimum cost over all states. The search may move only to another state that is 'close to' or 'in the neighborhood of' the current one, i.e. it is a *local* search. If V_{curr} is the current state vector, then the local neighborhood $Neighborhood(V_{curr})$ is the set of ϵ-vectors of size n obtained from V_{curr} by negating a single element (i.e. changing a 1 to a -1 or vice versa).

Accepting and Rejecting Moves. Simulated annealing combines hill-climbing with an ability to accept worsening state moves to provide for escape from local optima. From the current state V_{curr} a neighbouring state V_{neigh} is generated randomly. If $cost(V_{neigh}) < cost(V_{curr})$ then V_{neigh} becomes the current state (this is the 'hill-climbing', though perhaps 'valley diving' would be a better term for minimisation problems.) If not, then the state may be accepted probabilistically in a way that depends on the *temperature T* of the search (see below) and the extent to which the target state is worse (in terms of cost). The worse a target state is, the less likely it is a move to that state will be taken.

Cooling It All Down. In analogy with the physical annealing process, simulated annealing has a control parameter T, known as the temperature. Initially the temperature is high and virtually any move is accepted. Gradually the temperature is cooled and it becomes ever harder to accept worsening moves. Eventually the process 'freezes' and only improving moves are accepted at all. If no move has been accepted for some time then the search halts. We now describe the algorithm in detail.

The technique has the following principal parameters:

- the temperature T
- the cooling rate $\alpha \in (0, 1)$
- the number of moves N considered at each temperature cycle
- the number $MaxFailedCycles$ of consecutive failed temperature cycles
  (where no move is accepted) before the search aborts
- the maximum number IC_{Max} of temperature cycles considered before the search aborts

The initial temperature T_0 is obtained by the technique itself. The other values are typically supplied by the user. In the work described here they remain fixed during a run. More advanced approaches allow these parameters to vary dynamically during the search. The simulated annealing algorithm is as follows:

1) Let T_0 be the start temperature. Increase this temperature until the percentage of moves accepted within an inner loop of N trials exceeds some threshold (e.g. 95%).
2) Set $IC = 0$ (iteration count), $finished = false$ and $ILSinceLastAccept = 0$ (number of inner loops since a move was accepted) and randomly generate an initial current solution V_{curr}.
3) while(not $finished$) do 3a-3d
   a) **Inner Loop:** repeat N Times
      i) $V_{\text{new}} = \text{generateMoveFrom}(V_{\text{curr}})$
      ii) calculate change in cost
      $\Delta_{\text{cost}} = \text{cost}(V_{\text{new}}) - \text{cost}(V_{\text{curr}})$
      iii) If $\Delta_{\text{cost}} < 0$ then accept the move, i.e. $V_{\text{curr}} = V_{\text{new}}$
      iv) Otherwise generate a value u from a uniform(0,1) random variable. If $\exp^{-\Delta_{\text{cost}}/T} > u$ then accept the move, otherwise reject it.
   b) if no move has been accepted in most recent inner loop then
   $ILSinceLastAccept = ILSinceLastAccept + 1$
   else $ILSinceLastAccept = 0$
   c) $T = T * \alpha$, $IC = IC + 1$
   d) if (ILSinceLastAccept $>$ MaxFailedCycles) or $(IC > IC_{max})$ then
   $finished = true$
4) The state V_{best} giving the lowest cost over the whole search is taken as the final 'solution'.

Note that as T decreases to 0 then $\exp^{-\Delta_{\text{cost}}/T}$ also tends to 0 if $\Delta_{\text{cost}} > 0$ and so the chances of accepting a worsening move become vanishingly small as the temperature is lowered. In all the work reported here, the authors have used a value 0.95 for the geometric cooling parameter α and a value of 400 for N.

In attacking the PPP it will be useful also to record *when* each solution element (i.e. $V_0, \ldots, V_{n-1}$) changed for the last time. For current purposes, the time of last change to an element V_k is deemed to be the index IC of the inner loop in which the last change was made to that element (i.e. when a neighboring solution was obtained by flipping V_k and a move to that solution was taken). These times form the basis of the timing channel indicated earlier.

We shall show how heuristic search can be used to gain significant information on a problem. In fact, this information allows the problem to be reformalised as a reduced problem solvable by Grover's algorithm. First we describe the annealing attacks.

V. Attacking the Permuted Perceptron Problem

In 1999 Knudsen and Meier [6] showed that the $(m, n) = (101, 117)$ schemes recommended by Pointcheval were susceptible to a sophisticated attack based on an understanding of patterns in the results obtained during repeated runs of an annealing process. Essentially, their initial simulated annealing process is the standard one (with the number of trials at each temperature cycle equal to n) but with a modified cost function given by

$$Cost(V') = g\sum_{i=1}^{m}(max\{K-(AV')_i,0\})^R + \sum_{i=1}^{n}(|H(i)-H'(i)|)^R$$

$H(k) = \#\{j : (AV)_j = k\}$, i.e. the number of the $w_i = (AV)_i$ that have value k. H is the reference histogram for the target solution V (i.e. the histogram of the values in AV). Similarly H' is the histogram for the current solution V'. The histograms apply only to positive $(AV)_i$ elements. In all the experiments reported in [6] $R = 1.0$ and $K = 0$. There, repeated runs are carried out and commonality of the outputs from these runs is noted. Loosely speaking if all runs of the technique agree on certain secret element values there is a good chance that the agreed values are the correct ones. Agreed bits are fixed and the process carried out repeatedly until all bits agree. Unfortunately some (small number of) bits unanimously agreed in this way are actually wrong, and an enumerative search is made for these bits.

A. ClearBox Cryptanalysis - Looking Inside the Box

Virtually all applications of optimisation techniques in cryptography view optimisation as a black box technique. A problem is served as input, the optimisation algorithm is applied, and some output is obtained (a candidate secret in the PP and PPP examples). However, in moving from starting solution to eventual solution the heuristic algorithm will have evaluated a cost function at many (possibly hundreds of) thousands of points. Each such evaluation is a source of information for the guidance process. In the black-box approach this information is simply thrown away. For the PPP, the information loss is huge.

As the temperature cools in an application of simulated annealing it becomes more difficult to accept worsening moves. At some stage an element will assume the value of 1 (or -1) and then never change for the rest of the search, i.e. it gets stuck at that value. It is found that some bits have a considerable tendency to get stuck earlier than others when annealing is applied. (Indeed this observation is at the root of Chardaire et al.'s variant of annealing known as thermostatistical persistency [1].) One could ask 'Why?'. The answer is that the structure of the problem instance defined by the matrix and reference histogram exerts such influence as to cause this. The bits that get stuck early **tend to get stuck at the correct values**. Once a bit has got stuck at the wrong value it is inevitable that other bits will subsequently get stuck at wrong values too. However, it is unclear how many bits will get stuck at the right value before a wrong value is fixed. This has been investigated for various problem sizes and cost functions. Three problem sizes were considered as shown in Table I. For each problem size a cost function is defined by a value of g, a value of K and a value of R. Thirty problem instances were created for each problem size. For each problem and each cost function ten runs of the annealing process were carried out. The runs were assessed on two criteria: number of bits set correctly in the final solution and number of bits initially stuck correctly before a bit became stuck at an incorrect value.

(m,n)	Values of g_1	Values of K	Values of R
(101,117)	20,10,5	1,3,5,7,9,11,13,15	2,1.5,1
(131,147)	20,10	7, 10, 13, 16	2,1
(151,167)	20,15,10,5	5, 10, 15, 20	2,1

TABLE I
COST FUNCTION PARAMETER VALUES. ALL COMBINATIONS OF g, K AND R WERE USED.

Thus, for $(101, 117)$ instances there were $3\times8\times3 = 72$ cost functions and so 720 runs in total for each problem. The results are shown in Table II. For each problem the maximum number of correctly set bits in a final solution (i.e. the final result of an annealing run) is recorded together with the maximum number bits fixed correctly in a solution before a bit was set incorrectly (usually these will not be simultaneously achieved by one single solution).

B. Making Best Use of Available Information

Consider Ax for any solution vector x. Flipping any single element of x causes the components $(Ax)_i$ to change by ±2. Similarly, flipping any two bits of x causes the components to change by ±4 or else stay the same. Flipping three bits causes the components to change by ±2 or ±6. Generalising, if x may be transformed into the secret generating solution V by changing an even number of bits, then $(Ax)_i = (AV)_i \pm 4k$ for some integer k. Similarly, if an odd number of bit changes are needed then $(Ax)_i = (AV)_i \pm 4k + 2$. For any x let

$$SUMA(x) = \#\{i : (Ax)_i = 4k+1, for\ some\ k\}$$

$$SUMB(x) = \#\{i : (Ax)_i = 4k+3, for\ some\ k\}$$

$SUMA(V) = H(1) + H(5) + \ldots$ and $SUMB(V) = H(3)+H(7)+\ldots$ where H is the publicly available reference histogram. If V is obtained from x by an even number of bit changes, then we have $SUMA(V) = SUMA(x)$ and also $SUMB(V) = SUMB(x)$. If V is obtained from x by an odd number of bit changes, then $SUMA(V) = SUMB(x)$ and $SUMB(V) = SUMA(x)$. Only one of $SUMA(V)$ and $SUMB(V)$ can be odd (since their sum, n, is odd). Thus, for any vector x it is possible to determine whether it differs from V by an even or odd number of bits using the respective values of SUMA(x) and SUMA(V).

Suppose V is the actual secret and x is a solution obtained by annealing. If x is a high performing solution (with few bits wrong) then $(Ax)_i$ will typically be very close to $(AV)_i$. For the (101,117) problem instances, if $(Ax)_i = 1$ then the average actual value of $(AV)_i$ was 6.02. For (131,147) and (151,167) instances the averages were 6.23 and 6.46.

Suppose that $(Ax)_i = 1$ and ten bits are wrong. Typically it will be the case that $(AV)_i \in \{1, 5, 9, 13\}$. This observation has a big impact on enumerative search. For the sake of argument suppose that $(Ax)_i = (AV)_i = 1$. Then flipping the ten wrong bit values to obtain the actual secret must have no effect on the resulting value of $(Ax)_i$. This means that for five wrong bits we must have $a_{ij}x_j = 1$ and for the other

Prob	FBC	IBC
Pr 0	102	50
Pr 1	100	45
Pr 2	103	45
Pr 3	99	53
Pr 4	101	46
Pr 5	108	72
Pr 6	99	39
Pr 7	101	56
Pr 8	104	55
Pr 9	106	56
Pr 10	102	56
Pr 11	107	56
Pr 12	101	58
Pr 13	104	42
Pr 14	102	47
Pr 15	102	56
Pr 16	101	39
Pr 17	103	51
Pr 18	103	40
Pr 19	103	50
Pr 20	105	62
Pr 21	107	68
Pr 22	106	58
Pr 23	103	62
Pr 24	103	53
Pr 25	100	56
Pr 26	104	51
Pr 27	98	53
Pr 28	105	57
Pr 29	103	56
Size (101,117) 720 runs		

Prob	FBC	IBC
Pr 0	126	42
Pr 1	135	68
Pr 2	128	64
Pr 3	126	67
Pr 4	130	39
Pr 5	131	70
Pr 6	126	47
Pr 7	128	56
Pr 8	123	52
Pr 9	139	75
Pr 10	129	51
Pr 11	123	48
Pr 12	134	57
Pr 13	132	62
Pr 14	124	37
Pr 15	122	59
Pr 16	124	41
Pr 17	121	42
Pr 18	130	62
Pr 19	129	53
Pr 20	132	67
Pr 21	128	59
Pr 22	129	97
Pr 23	127	61
Pr 24	126	43
Pr 25	127	72
Pr 26	132	44
Pr 27	125	68
Pr 28	126	38
Pr 29	123	50
Size (131,147) 160 runs		

Prob	FBC	IBC
Pr 0	148	72
Pr 1	142	64
Pr 2	145	66
Pr 3	157	88
Pr 4	147	58
Pr 5	140	67
Pr 6	151	86
Pr 7	135	48
Pr 8	143	55
Pr 9	150	95
Pr 10	149	61
Pr 11	145	70
Pr 12	143	49
Pr 13	138	63
Pr 14	147	58
Pr 15	141	63
Pr 16	151	56
Pr 17	144	82
Pr 18	147	98
Pr 19	137	47
Pr 20	136	69
Pr 21	140	59
Pr 22	142	55
Pr 23	146	67
Pr 24	138	69
Pr 25	147	69
Pr 26	145	61
Pr 27	146	68
Pr 28	141	64
Pr 29	143	80
Size (151,167) 320 runs		

TABLE II

MAXIMUM FINAL BITS CORRECT (FBC) AND MAXIMUM INITIAL BITS CORRECT (IBC) OVER ALL RUNS. TOTAL NUMBER OF RUNS SHOWN FOR EACH PROBLEM SIZE. THIRTY PROBLEM INSTANCES WERE ATTACKED FOR EACH PROBLEM SIZE.

five we must have $a_{ij}x_j = -1$. This reduces any enumerative search. For example, searching over 117 bits would usually require C_{10}^{117} (around 4.4×10^{15}) but now requires a search of order around $C_5^{58} \times C_5^{57}$ (around 2.1×10^{13}). This assumes that for solution x $\#\{x_j : a_{ij}x_j = 1\} = 58$ and $\#\{x_j : a_{ij}x_j = -1\} = 57$ (or vice versa). In practice, this may not be the case but any skew actually reduces the complexity of the search. In this respect, it may be computationally advantageous to consider some $(Ax)_i < 0$. For example, if $(Ax)_i = -7$ and there are 10 bits wrong then $(AV)_i$ must be in the range 1..13 with the smaller values much more likely. If $(AV)_i = 1$ then there must be seven wrong bits currently with $a_{ij}x_j = -1$ and three with $a_{ij}x_j = 1$. This is a powerful mechanism that will be used repeatedly.

One has to guess the relationship of $(Ax)_i$ to $(AV)_i$. This will generally add only a factor of about four to the search (and often less). One has also to determine how many bits are actually wrong too. One can start by assuming that the solution vector has the minimum number of bits wrong yet witnessed and engage in enumerative searches. If these fail, simply increment the number of bits assumed incorrect by 2 and repeat the search processes (only even numbers or odd numbers of wrong bits need be considered). The complexity of the search is dominated by the actual number of wrong bits (searches assuming fewer numbers of wrong bits are trivial by comparison). The complexities reported in this paper therefore assume knowledge of the number of wrong bits in the current solution.

C. The Direct Attack

It is obvious that 'warping' the cost function produces results that are indeed better than those obtained under the natural cost function. Thus, in the $(101, 117)$ problems three (5, 11 and 22) have given rise to solutions with 10 bits or fewer wrong (from the FBC column of Table II). Once the highest performing solution has been selected (a factor of 720) an enumerative search of order $C_5^{58} \times C_5^{57}$ (which is less than 2^{45}) will find the solution in these cases. For the $(131, 147)$ and $(151, 167)$ instances extreme results are also occasionally produced. (131,147) Problem 9 gave rise to one solution with only 8 bits wrong. (151,167) Problem 3 similarly gave rise to a solution with only 10 bits wrong. This would require a total search of approximately $320 \times C_5^{84} \times C_5^{83}$ which is less than 2^{60}. This is not the most efficient way of solving the problem however.

VI. APPLYING QUANTUM SEARCH TO THE RESULTS OF ANNEALING ATTACKS

In the above simulated annealing has made major progress in solving the PPP but cryptography does not accept 'near' solutions. If you haven't solved a problem - you simply haven't solved it. In some cases the problem is brought within computation reach but in many ot simply remains unbroken. However, armed with the above results we can see two clear ways of applying Grover's algorithm:

Problem	Greatest	Average	Least
(101,117) No. Bits Remaining	19	114.2	9
Quantum Search Complexity	2^{67}	2^{53}	2^{32}
(131,147) No. Bits Remaining	26	19.2	8
Quantum Search Complexity	2^{104}	2^{76}	2^{32}
(151,167) No. Bits Remaining	32	22.9	10
Quantum Search Complexity	2^{128}	2^{92}	2^{40}

TABLE III
QUANTUM RESIDUAL INDEX SOLUTIONS (INDEX LENGTH = 7 BITS FOR (101,117) AND 8 FOR OTHERS

Problem	Greatest	Average	Least
(101,117) No. Bits Remaining	78	64.1	49
Quantum Search Complexity	2^{39}	2^{32}	2^{25}
(131,147) No. Bits Remaining	110	90.6	50
Quantum Search Complexity	2^{55}	2^{46}	2^{25}
(151,167) No. Bits Remaining	120	100.1	69
Quantum Search Complexity	2^{60}	2^{50}	2^{35}

TABLE IV
QUANTUM RESIDUAL TRAJECTORY SOLUTIONS

- Trajectory reduced attack: assume at least the 'initial' R bits are correct and use Grovers to search over all remaining bits. This has state spaces 2^{117-R}, 2^{147-R} and 2^{167-R} respectively. By Grovers search a solution is findable in $O(2^{\frac{117-R}{2}})$, $O(2^{\frac{147-R}{2}})$ and $O(2^{\frac{167-R}{2}})$.
- Index attack: assume W wrong bits. Then assuming an index can be expressed in k bits, our state space for the indices of the incorrect bits is of size $k * W$. By Grover's algorithm this can be achieved by a search of $O(2^{\frac{k\times W}{2}})$ operations. For indices less than 255 eight bits suffices to encode an index. Thus, Grover's search gives now a search complexity of $O(2^{\frac{8\times W}{2}})$ operations. For the (101,117) case 7 bits will suffice.

For the three problem cases the Table III records the quantum computational complexity for the worst, average and best cases of the direct annealing attacks. IV records the quantum computational complexity for the worst, average and best cases of the annealing trajectory attacks.

We can see that almost **all** $(101, 117)$ direct attack reduced problems are brought within quantum computational range. For the $(131, 147)$ case only the best case seems obviously feasible (but in this case it is well within range). For the $(151, 167)$ case again only the best case result seems feasible.

However, when we look at the trajectory reduced problems, it would appear that **all** problem instances for all sizes can be attacked successfully.

VII. WHAT DOES THIS SHOW?

The above results show that real cryptographic problems can be attacked by heuristic search and by quantum search but that fusing the two techniques can deliver results unattainable by either: the application of quantum search to the trajectory results of annealing searches suffices to break all instances of all considered problem types.

After significant computational efforts it appears that annealing is unable to get the right answer on its own: the nature of underlying problem is just too non-linear. However, annealing can be used to gain *significant information* on the underlying solution. This has produced a reduced problem that is attackable by quantum means.

This suggests a new way of thinking about the use of nature inspired search. Currently the nature-inspired cryptology community spends all of its time trying to *solve*the direct problems it is set. We believe that in many cases the nature of the problem will simply defeat such attempts: crypto-systems are often designed to by extraordinarily non-linear, or at least sufficiently non-linear to defeat guided search attacks. In a sense, we may often be destined to fail by design (of the crypt algorithm designer). However, if we reduce our goals of the search we may well be able to find answers to simpler problems. We believe that the guideline will prove of very significant use our community:

The Information Gain Principle: The purpose of applying nature-inspired search to cryptanalysis problems should be to gain information on the underlying solution.

This has implications for the problems we actually set for nature-inspired searches. If we can identify a reduced problem that is solvable by another technique, then we should target that reduced problem (and choose a cost function appropriately).

Note that many of the cost functions used above were not intended to 'solve' the underlying original problem. The warped functions were an innovative attempt to 'see what would happen of'. Generally optimal values of the cost functions used are not obtained by solutions to the underlying problem. The aim was to maximise the information gain; this information could then be further exploited by other , more appropriate means, be that brute force classical enumeration or Grover's quantum search.

REFERENCES

[1] P Chardaire, J C Lutton, and A Sutter. Thermostatistical Persistency: A Powerful Improving Concept for Simulated Annealing. *European Journal of Operations Research*, 86:565–579, 1995.

[2] D.E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley, 1989.

[3] L.K. Grover.

[4] Giddy J.P. and Safavi-Naini R. Automated Cryptanalysis of Transposition Ciphers. *The Computer Journal*, XVII(4), 1994.

[5] S. Kirkpatrick, Jr. C. D. Gelatt, and M. P. Vecchi. Optimization by Simulated Annealing. *Science*, 220(4598):671–680, May 1983.

[6] Lars R. Knudsen and Willi Meier. Cryptanalysis of an Identification Scheme Based on the Permuted Perceptron Problem. In *Advances in Cryptology Eurocrypt '99*, pages 363–374. Springer Verlag LNCS 1592, 1999.

[7] Robert A J Mathews. The Use of Genetic Algorithms in Cryptanalysis. *Cryptologia*, XVII(2):187–201, April 1993.

[8] David Pointcheval. A New Identification Scheme Based on the Perceptron Problem. In *Advances in Cryptology Eurocrypt '95*. Springer Verlag LNCS X, 1995.

[9] Eleanor G. Rieffel and Wolfgang Polak. An Introduction to Quantum cComputing for Non-Physicists. *ACM Computing Surveys*, 32(3):300–335, 2000.

[10] Richard Spillman, Mark Janssen, Bob Nelson, and Martin Kepner. Use of A Genetic Algorithm in the Cryptanalysis of Simple Substitution Ciphers. *Cryptologia*, XVII(1):187–201, April 1993.

[11] Jaques Stern. A New Identification Scheme Based On Syndrome Decoding. In *Advances in Cryptology —Crypto '93*, pages 13–21. Springer Verlag LNCS 773, 1997.

[12] Forsyth W.S. and Safavi-Naini R. Automated Cryptanalysis of Substitution Ciphers. *Cryptologia*, XVII(4):407–418, 1993.