Integer Arithmetic without Arithmetic Addition

Applications: Discriminant Analysis, Cryptographic Primitives

Gideon Samid
Department of Electrical Engineering and Computer Science
Case Western Reserve University
Gideon.Samid@CASE.edu

Abstract: Revisiting long established conventions has proven very fertile in many a case. Let's then revisit the premise that arithmetic must be constructed with the arithmetic addition as its foundation. Here we explore an arithmetic realm over integers without invoking the quintessential operation of addition. We propose an arithmetic constructed over a fundamental mapping of one set of integers into another. We start and focus here on mapping an arbitrary number of integers to a single integer, and further limit our investigation to a mapping procedure that views the input integers as a set of conflicting answers to a binary question, and attempt to figure out the single integer that best reflects the combined "wisdom" of the input answers. Thereby we construct the proposed arithmetic as ground tool for discriminant analysis. On the other end, the many-to-one mapping suggests this arithmetic as a fundamental hashing function, and the complexity of data loss suggests a new primitive for asymmetric cryptography. This arithmetic evolved from practical algorithms used by the author in his engineering practice, where the original name was BiPSA: Binary Polling Scenario Analysis. For continuity purposes we carry on the name. This article focuses on the skeleton arithmetic. Applications and substantiation will follow.

1.0 Introduction

The arithmetic addition belies the subsequent range of mathematical operations: subtraction, multiplication, division, raising to a power, etc. Just for that reason alone one should submit to the curiosity of how would one construct an arithmetic without invoking the operation of adding two quantities to measure their combination. Given that the natural numbers are sequentially defined with it (“+1”) this premise seems too ambitious. But then the series of natural numbers can be defined as follows: given n numbers, we can define as the next number (marked as “n+1” for convenience) as a number which is not 1, not 2, not 3….., not n. Thereby leaving the ‘distance’ between the successive numbers undefined. The various numbers simply indicate the order in which they were included in the set of counted natural numbers. An arithmetic without ‘addition’ will then rely on the mere distinction between the numbers, (and their natural order), without quantifying that distinction. An attempt at such a construction was reported last year [Samid-10], and in due course it became clear that the practical discriminant analysis employed by the author in practice also so qualifies, and such is of interest beyond its immediate application.

The roots of the proposed arithmetic are in a procedure known as BiPSA -- Binary Polling Scenario Analysis, which was developed as a discriminant analysis tool where any number of opinion sources, *n*, offered their opinion (vote) over a binary question (yes/no), and the objective was to integrate the various opinions (votes) into a single opinion, which is expected to be a "fair representation" of the "summary wisdom" of the *n* voters. The idea was to construct a unit integrator that would regard all voters as having equal impact on the result, and capture the relative significance of each voter through a network configuration of these unit integrators, which would be easy to manipulate. The voters would indicate their binary opinion along with a measure of self-confidence in their own answer. The opinion and the confidence measure would determine the output of the BiPSA integrator. The success of this construction suggested a formal definition as described herewith. The name BiPSA indicates that the original use of BiPSA was for analyzing a projected scenario with the binary question: *is that scenario, as described, more likely to happen, or more likely not to happen?*

2.0 The Basic BiPSA Algebra Concept

We define a "BiPSA (n/N) set" (= $\mathbf{B_p}(n/N)$) as a set of n integers, $b_1, b_2, \ldots. b_n$ each in the range {-N:+N}. "*n*" will be referred to as the "BiPSA count" and "N" will be the BiPSA range.

We define a "→" as the BiPSA operation mapping Bp(n/N) to Bp(m/N)

$$\mathbf{B_p}(n/N) \rightarrow \mathbf{B_p}(m/N)$$

where *n*, and *m* are arbitrary natural numbers.

Equivalently:

$$\mathbf{B_p}(m/N) = ß\ \mathbf{B_p}(n/N)$$

The symbol "ß" will designate the BiPSA operator. The BiPSA operator maintains the BiPSA range.

We define the set of all BiPSA sets, as the **BiPSA realm** . Any BiPSA operator maps an element of the BiPSA realm to the same or another set of same realm. We may write:

$$\mathbf{B_p}(m/N) = ß_k ß_{k-1} \ldots. ß_1 \mathbf{B_p}(n/N)$$

interpreted as the BiPSA set $\mathbf{B_p}(n/N)$ is operated on with $ß_1$, and the resultant BiPSA set is operated on with $ß_2$; similarly the resultant BiPSA set is operated on wtih $ß_3$, and so on until $ß_k$, resulting in $\mathbf{B_p}(m/N)$.

If a certain ß operator operates on a set $\mathbf{B_p}$(n/N) *t* times then we may write:

$$\mathbf{B_p}(m/N) = ß^t\mathbf{B_p}(n/N)$$

We now define the "**BiPSA Unit Operator**" (BiPSA-UI, or ßUI) as a BiPSA operator that outputs a BiPSA set containing one member:

$$\mathbf{B_p}(1/N) = ß_{UI}\mathbf{B_p}(n/N)$$

Alternatively we shall write:

$$\mathbf{B_p}(1/N) = b_0= [b_1, b_2.......b_n]$$

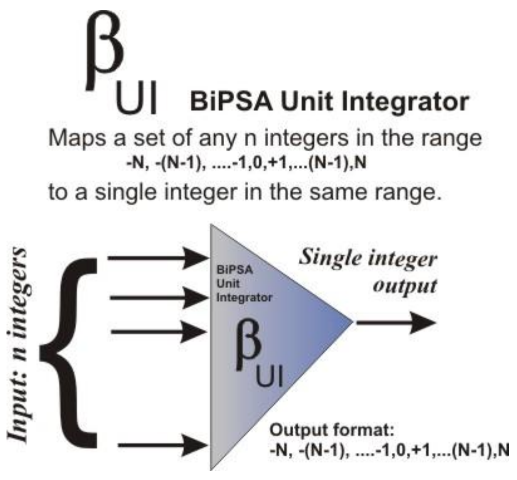
using the subscript "0" to denote the output integer of the BiPSA unit integrator, and the squared brackets to indicate the BiPSA unit operation.

The BiPSA unit operator is called normal if it is consistent with the following terms of constraint:

- **1. Permutation invariance.**
- **2. Symmetry**
- **3. Monotony**
- **4. Range limits**
- **5. Full-range terms for same sign instances.**
- **6. Full-range terms for mixed signs instances.**
- **7. N-invariance**

Explained:

permutation: This term is really self evident because the $ß_{UI}$ operates on a set which is an unordered collection of elements. This attribute is emphasized with reference to the graphic, "wired" depiction of the ßUI, which is to say that the various integer values can switch their wire location without affecting the output.


β_{UI} BiPSA Unit Integrator
Maps a set of any n integers in the range
-N, -(N-1), ...-1,0,+1,...(N-1),N
to a single integer in the same range.
Input: n integers
BiPSA Unit Integrator
β_{UI}
Single integer output
Output format:
-N, -(N-1), ...-1,0,+1,...(N-1),N

symmetry: If all the signs of the *n* input integer variables are switched, then the sign of the BiPSA output switches sign too, but the absolute value remains the same. Say: Let $b^*_i = -b_i$ for *i=1,2,3,...n* then: $[b_1, b_2, b_3, ...b_n] = -[b^*_1, b^*_2, b^*_3, ...b^*_n]$

monotony: The monotony term says that if the value of an input variable increases, (while the other values stay unchanged) then the $ß_{UI}$ output value must not decrease. It can increase or stay the same. And conversely, if the value of an input variable decreases, then the $ß_{UI}$ output value must not increase

range limits: If M is the largest absolute value of any input to the $ß_{UI}$, then the absolute value of the output will not be larger than M.

full-range terms for same sign instances: This term refers to BiPSA input sets where the input integers have no conflicting signs (either all the signs are positive, and zero, or all the signs are negative and zero). It specifies that among all the possible $\mathbf{B_p}(n/N)$ that may be processed by the $ß_{UI}$ (there are $(N+1)^n)/n!$ sets like that), there would be at least one such set that maps into the output value of 1, and at least another set that maps into the output value of 2, one for 3, etc., up to N. In other words, it would be possible to produce the output range of {-N:+N} from same sign input sets.

full-range terms for mixed sign instances: this term specifies that among all the possible $\mathbf{B_p}(n/N)$ input sets to the $ß_{UI}$, where one or more integers is positive, and one or more integers is negative, there will be at least one such input set where the output would be -(N-1), and one set where the output would be -(N-2), etc, through 0, and to an output of (N-1). In other words it would be possible to produce the output range of {-(N-1):+(N-1)}, from mixed-sign input sets.

N-invariance: This term specifies that for any given input set, the output would be the same regardless of the limiting value N for the set (provided N is higher than any absolute value of the set). To wit:

$$[\mathbf{B}_\mathbf{p}(n/N)] = [\mathbf{B}_\mathbf{p}(n/M)]$$

for any M>N. So that the $ß_{UI}$ output for the input set [5,-3, +201, -14] would be the same whether the value of N (the maximum allowed absolute value for input) is N=201, or N=202, or higher.

Below we first present some BiPSA theorems, and then define the The Proper BiPSA Unit Integrator.

2.1 Properties of Nominal BiPSA Unit Integrators

The following theorems hold true for any nominal BiPSA unit integrator:

T-1. 0 = [0,0,...0]

Or say, if all i=*1,2,3....n* $b_i = 0$ then $b_o = 0$.

Proof: This is due to the symmetry term. If (x ≠ 0)= [0,0,...0] then upon reversal of signs the output should be -x, while the input variables do not change, which is an impossibility according to the single output term.

T-2: For every BiPSA range N and BiPSA count n, if b_i= M for all i=1,2,3...n, then b_o= M

Or say:

$$M = [M,M,M,......M]$$

Proof: Suppose we had: (X ≠ M) =[M,M,...M], then if the respective range, N=M, we would have X < M. (Since the output range would be M:-M). And the output of M would have to be generated by another input set (due to the full range term): M = [b_1, b_2, b_3, ...b_n]

where one of the inputs, at least, is smaller than M, which will violate the monotony term, and hence, necessarily X=M.

T-3: M-1 = [M,-1]

Proof: If M is also the range, N=M, then the pair (M,-1) is the highest mixed signs combination, and thus to satisfy the monotony term and the full range term it sould evaluate to yield the highest BiPSA output value in the range 0,...(M-1). And because of the N-invariance term, the same is true for any value of M ≤N.

T-4: {M-1,M} = [M,L] for M >L ≥ 0

Proof: X= [M,L] cannot be smaller than M-1 because of T-3, and the monotony term; it cannot be larger than M because of the N-invariance term, and hence the only allowable values for X are M-1, and M.

note: the above four theorems are accompanied by symmetric four theorems where all integers are of opposite signs. The latter are proven by the symmetry terms..

2.2 The Proper BiPSA Unit Integrator

We shall now define a normal BiPSA unit operator which we will designate as 'proper':

A proper BiPSA unit operator is a normal unit operator which also satisfies the following proprietary terms:

- The binary same signs term
- The binary mixed signs term
- The binary zero term
- The BiPSA Count Reduction Procedure

defined as follows:

The Binary Same Sign Term: states that:

M-1 = [M, L] for L=1,2,....(M-2) and M = [M, L] for L=M-1   This further specifies and is consistent with theorem T-4.

The Binary Mixed Signs Term: states that:

(M-L) = [M,-L] For L and M natural numbers and M >L

The Binary Zero Term: states that:

for M>1 [M,0]=M-1, and [1,0]=1

The above three terms fully define the proper BiPSA unit integrator for BiPSA count n=2.

For n=1 the invariance terms requires that: M=[M], for any integer M.

Note: while the specified BiPSA terms seem to involve subtraction (reverse addition), this is only for convenience. The expression, say: M-L (M,L natural numbers) may be interpreted as counting down the numbers list M-1, M-2, L times. It is different from the classic notion of addition, amassing say B units on top of A units to a total quantity of A+B.

We shall now define the Reduction to Binary procedure that would convert any BiPSA set with n>2 to a binary BiPSA set with 1 ≤n ≤2.

2.2.1 The BiPSA Count Reduction Procedure

The basic procedure is structured as follows:

1. Filter out the negative integers, then reduce the resultant set to a single non-negative integer.
2. Filter out the positive integers, then reduce the resultant set to a single non-positive integer
3. Resolve the results of the (1) and (2) according to the binary solutions above.

Filtering is the process of replacing the filtered integer with a zero ("+0"). The reduction of the 'same sign' set (the filtered sets) proceeds as follows:

Same sign reduction procedure: based on iterative reduction of the size of the BiPSA input set, until n=1. The description below applies for the non-negative set. A symmetric procedure will apply to the non-positive set.

We disinguish between an even number of BiPSA inputs and an odd number. For an odd number (n=2k+1): Let $b_1 \le b_2 \le b_3 \le .....\le b_{2k+1}$ be a sorted ascending list of the BiPSA input values. Input b_{k+1} is the middle of that list. That value is put aside, to start a second list, $b'_1=b_k$. The resultant original list now is left with 2k elements: $b_1, b_2,... b_k, b_{k+2}.....b_{2k+1}$
We now remove the two center elements: b_k, b_{k+2}, and compute their BiPSA value according to the procedure for two BiPSA elements: $b'_2= [b_k, b_{k+2}]$ and define the result as the second element of the second list. The resultant original list now is: $b_1, b_2,... b_{k-1}, b_{k+3}.....b_{2k+1}$
Again, we remove the central pair and use the two-elements BiPSA procedure to define the next element in the second BiPSA list: $b'_3= [b_{k-1}, b_{k+3}]$

We repeat this procedure, each round we remove the central pair in the original BiPSA list, and compute its BiPSA value, then add that value to the second BiPSA list. We continue with these rounds until we exhaust the entire first BiPSA list.

If the original BiPSA list is comprised of an even number of elements: n=2k, then we apply the same procedure. Here too, we arrange the BiPSA set in ascending order: $b_1 \le b_2 \le b_3 \le$

.....$\leq b_{2k.}$ The difference is that in this case the first element in the second BiPSA list is: $b'_1= [b_{k-1}, b_{k+1}]$, followed by: $b'_2= [b_{k-2}, b_{k+2}]$, and so on until $b'_k= [b_1, b_{2k}]$

The second BiPSA list is of length n/2 if the original list is even, and of length (n+1)/2 if the original list is odd. We now treat that second list, as if it were the original BiPSA list and thereby we generate a third BiPSA list, which is again half size, or nearly half size of the secondary BiPSA list. The so generated third BiPSA list now takes on the role of the original BiPSA list: namely, it generated a fourth BiPSA list of half size or nearly so. This iterative procedure continues until the last generated BiPSA list is of size one which is to be regarded as the BiPSA evaluation of the filtered BiPSA list, or say the single integer reduction of the filtered list.

Symbolically for the odd case: The original list: $b_1, b_2,... b_k, b_{k+2}.....b_{2k+1}$
generates a secondary list: $b_{k+1}, [b_k, b_{k+2}], [b_{k-1}, b_{k+3}], \ldots [b_{k-i}, b_{k+i+2}], \ldots [b_1, b_{2k+1}]$. which is served as input to the same process, and so repeatedly until the size of the output list is one. If the original list has even numbers: $b_1, b_2,... b_k, b_{k+2}.....b_{2k}$
Then the secondary list is defined as: $[b_k, b_{k+1}], [b_{k-1}, b_{k+2}], \ldots [b_{k-i}, b_{k+i+1}], \ldots [b_1, b_{2k}]$.

Thereby we have now fully defined the BiPSA operation (β_{UI}) over any arbitrary list of integers, identifying the integer that BiPSA represents the full list.

To illustrate let's compute the BiPSA value of : **[4, 0, -1, 2, -2, 1, -3].** The non-negative set, will be: [4,0,0,2,0,1,0] , sorted: [0,0,0,0,1,2,4] . We now build the second BiPSA list: 0, [-

0,1], [0,2], [0,4] = 0, 1,1,3 from which we build the third list: [1,1], [0,3] = 1, 2, which evaluates to: [1,2]=2. The non-positive set is: [0,0,-1,0,-2,0,-3], sorted: [0,0,0,0,-1,-2,-3]. We now build the second BiPSA list: 0,[0,-1],[0,-2],[0,-3]=0,-1,-1,-2, from which we build the third list: [-1,-1], [0,-2]=-1,-1 which evaluates to [-1,-1]=-1. We now BiPSA evaluate the non-negative reduction result against the non-positive reduction result: [2,-1]=1, and hence: 1 = [4, 0, -1, 2, -2, 1, -3]

Second illustration: **? = [-12, 1, 11, 4, 3, -7, 4, 0].** The non-negative list: [0,1,11,4,3,0,4,0], sorted: [0,0,0,1,3,4,4,11]; the second list: 3,[1,4],[0,4],[0,11]=3,3,3,10; the third list: [3,3],[3,10]=3,9 which evaluates to [3,9]=8. The non-positive list: [-12,0,0,0,0,-7,0,0] sorted: [0,0,0,0,0,0,-7,-12]; the second list: 0,[0,0],[0,-7],[[0,-12]= 0,0,-6,-11; the third list: [0,-6],[0,-11]=-5,-10 which evaluates to -9, and hence: = [-12, 1, 11, 4, 3, -7, 4, 0]=[8,-9]=-1

3.0 Vector and Matrix Algebra (BiPSA)

We shall advance the basic BiPSA concept to vector and matrix algebra.

We cover the following topics:

- zero notations
- vector algebra
- matrix algebra

Zero Notations

We shall distinguish between a minus zero (-0) and a (+0) as BiPSA elements. To understand these notations we need to resort to the wired diagram where the BiPSA unit operator is depicted as a logical entity that receives *n* inputs (the BiPSA elements), and generates a single output (the BiPSA result). If a given input entry is not 'charged' with an input value (an integer), then this absence of data will be noted as minus zero, -0. If that input value (wire) is charged with the integer zero as input then it would be designated as plus zero, (“+0”).

Hence: [9,4,-3,-7,5,12,-3,7] =[9,4,-0,-3,-7,5,-0,-0,12,-3,7] ⍰ ≠ [9,4,0,-3,-7,5,0,0,12,-3,7]

Vector Algebra

Owing to the N-invariance condition, it is not necessary to specify the BiPSA range, N, (when we limit our interest to proper integration), since the result will be the same regardless of the value of N. So:

$$\mathbf{B_p}(n/N) = \mathbf{B_p}(n)$$

We can now reduce the BiPSA set to an ordered list of the BiPSA integers (without affecting its integrated result), and thereby qualify this ordered list as a 'vector'. A BiPSA set so defined will be designated as **B**.

Two BiPSA vectors of same length (same BiPSA count, n) may be added as follows:

$$\mathbf{B}_{a+c}= \mathbf{B}_a+ \mathbf{B}_c= [(b_{a1}+ b_{c1}), (b_{a2}+ b_{c2})\text{.....}[(b_{an}+ b_{cn})]$$

This addition is auxiliary, and not fundamental to the BiPSA arithmetic, and thus the characterization of BiPSA as an arithmetic without the arithmetic addition still holds. Alas, the ‘theologians’ among us may either disregard the operation of addition as defined above, or dispute the title of the BiPSA arithmetic. Either way it does not detract from the thesis as a whole.

In general:

$$ß_{UI}(\mathbf{B}_a+ \mathbf{B}_c) \neq ß_{UI}\mathbf{B}_a+ ß_{UI}\mathbf{B}_c$$

For example: let $\mathbf{B}_a$= [2,-0,4] and $\mathbf{B}_c$= [-2,3,1]: We compute: Since [2,0,4] + [-2,3,1] = [0,3,5], and find: $ß_{UI}$[2,0,4]=2, $ß_{UI}$[-2,3,1]=2, and $ß_{UI}$[0,3,5]=4, so here [2,-0,4]+[-2,3,1]=[0,3,5]=4 . However, if we check: $\mathbf{B}_a$= [-4,-1,2]; $\mathbf{B}_c$= [0,3,-4]: and compute: [-4,-1,2] + [0,3,-4] = [-4,2,-2], then $ß_{UI}$[-4,1,2]=2, $ß_{UI}$[0,3,-4]=0, and $ß_{UI}$[-4,2,-2]=-1 we find that: [-4,-1,2] + [0,3,4] ≠ [-4,2,-2].

We may also define multiplication of a BiPSA vector with a scalar in the form of an integer *t*:

$$t[b_1,b_2,....b_n] = [tb_1,tb_2,....tb_n]$$

which may lead to a variety of expressions, and equations in the form:

$u\mathbf{B}_a + v\mathbf{X} = \mathbf{B}_c$ where *u* and *v* are integers, and **X** is a BiPSA vector that satisfies the above equation, B_a and B_c are given BiPSA vectors. We may also specify a BiPSA vector **X** such that:

$$ß_{UI}\mathbf{B}_a + ß_{UI}\mathbf{X} = ß_{UI}\mathbf{B}_c$$

and similarly write a variety of BiPSA equations for which the existence, the number, and the finding of proper solutions for X is an expected mathematical challenge. A system of equations with several BiPSA vectors unknown is readily extended to.

3.1 Vector Multiplication

By its definition any ordered list of integers will qualify as a vector in the BiPSA algebra, and may be BiPSA evaluated. Let $\mathbf{W}= [w_1,w_2,....w_n]$ be such a vector while $\mathbf{B}= [b_1,b_2,....b_n]$ will be an ordinary BiPSA set written as a vector. We shall now define a vector multiplication:

$$b_{B,W} = \mathbf{B} * \mathbf{W} = [b_1w_1; b_2, w_2; .....b_nw_n] = [b_1, b_2, ...b_n]*[w_1, w_2, ...w_n]$$

resulting in the integer $b_{B,W}$.

The vector multiplication algorithm operates as follows:

For m=1,2,...w_{max} let's define $\mathbf{B}^m$ as a BiPSA set constructed as follows:

$$\mathbf{B}^m = (\delta_1\lambda_1 b_1, \delta_2\lambda_2 b_2, \ldots\ldots\ldots\ldots \delta_n\lambda_n b_n)$$

where if $|w_i| \geq m$, then $\delta_i = 1$, otherwise: $\delta_i = 0$. And if $w_i \leq 0$ then $\lambda_i = -1$, otherwise $\lambda_i = 1$ and where w_{max} is defined as: $w_{max} = \mathrm{Max}(|w_1|, |w_2|, \ldots.|w_n|)$.

The w_{max} BiPSA entities ($\mathbf{B}^m$) will be BiPSA operated, resulting in w_{max} integers, which in turn will be grouped to a new BiPSA entity, get BiPSA operated on, and yield the integer value of $b_{B,W}$. Namely:

$$b_{B,W} = ß_{UI}(ß_{UI}(\mathbf{B}^1), ß_{UI}(\mathbf{B}^2) \ldots\ldots\ldots ß_{UI}(\mathbf{B}^{w}{}_{max}))$$

Illustration of vector multiplication:

Let $\mathbf{B}$ = [3,-1,2,+0,4] and $\mathbf{W}$ = [3,0,-2,-2,1]. So n=5. We shall now construct the $\mathbf{B}^m$ entities: $w_{max} = 3$, so:

$\mathbf{B}^1$= [1*1*3, 0*0*(-1), 1*(-1)*2, 1*(-1)*0, 1*1*4]=[3,-2,0,4]

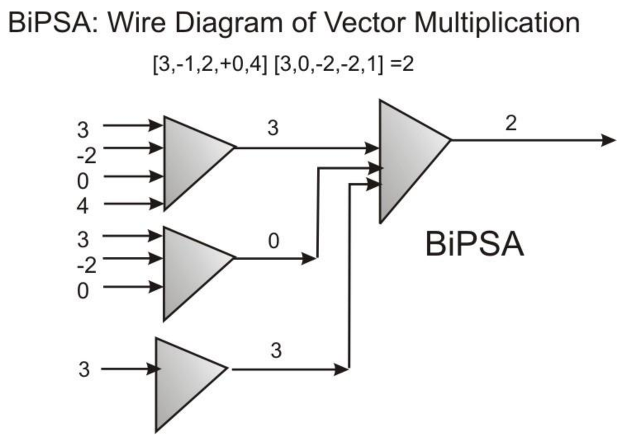
$\mathbf{B}^2$= [1*1*3, 0*0*(-1), 1*(-1)*2, 1*(-1)*0, 0*1*4]=[3,-2,0]

$\mathbf{B}^3$= [1*1*3, 0*0*(-1), 0*(-1)*2, 0*(-1)*0, 0*1*4]=[3]

And hence: b_{BW}= [[3,-2,0,4],[3,-2,0],[3]]=[3,0,3]=2=[3,-1,2,+0,4] [3,0,-2,-2,1] .

This illustration also clarifies that if $\delta=0$, it removes its member from the list, same in the case where a member of the BiPSA set equals "-0", but not so when the member of the BiPSA set equal "+0" .

Wire Diagram of Vector Multiplication:



BiPSA vector multiplication may be depicted graphically through a "wire diagram" that connects BiPSA unit integrators. Every BiPSA value shows up in this diagram a number of times, where this number is given by the absolute value of the corresponding integer in the **W** vector. If the sign of that integer is negative, then it flips the result of the respective BiPSA unit integrator. The above illustration is wire-diagramed herein.

Properties of Vector Multiplication

Vector multiplication is not commutative. In general:

$$\mathbf{B} * \mathbf{W} \neq \mathbf{W} * \mathbf{B}$$

Let us define the unit **W** vector as: ε = [1,1,1,......1],

so that: t*ε = [t,t,t,.....t] for any integer t. We can now write the epsilon multiplication [Eq. v-1] property:

$$ß_{UI}(\mathbf{B} *(t* \varepsilon)) = ß_{UI}(\mathbf{B})$$

proof: for **W**=ε, we have:

$$b_{0\varepsilon}= ß_{UI}\mathbf{B} = ß_{UI}\mathbf{B}^1= ß_{UI}\mathbf{B}^2= ......= ß_{UI}\mathbf{B}^n$$

And hence:

$$b_{0\varepsilon}= [b_0, b_0, ...b_0]= b_0$$

which proves the epsilon multiplication property .

3.2 Matrix Operations

Let's consider m vector multiplications:

$\mathbf{B} * \mathbf{W}_1$
$\mathbf{B} * \mathbf{W}_2$
....
....
$\mathbf{B} * \mathbf{W}_m$

The m **W** vectors can be structured into a matrix, column wise:

Using the notation: $\mathbf{W}_i = (w_{i1}, w_{i2}, \ldots\ldots. w_{in})$:

$$\Omega = \begin{bmatrix} w_{11} & \cdots & w_{m1} \\ \vdots & \ddots & \vdots \\ w_{1n} & \cdots & w_{mn} \end{bmatrix}$$

Leading to a newly defined operation of a BiPSA vector multiplied by the Ω matrix:

$$\mathbf{B} * \Omega = \mathbf{B'}$$

where **B'** is a BiPSA vector of count n' =m, equal to the number of columns in the Ω matrix.

Let's designate the count (number of elements) of a BiPSA vector by 'absolute value' markings: $n = |\mathbf{B}|$; $n' = |\mathbf{B'}|$. Accordingly, we will identify three vector-matrix multiplication modes, using cryptographic nomenclature:

Ω: $|\mathbf{B}| > |\mathbf{B'}|$ hashing
Ω: $|\mathbf{B}| = |\mathbf{B'}|$ nominal encryption
Ω: $|\mathbf{B}| < |\mathbf{B'}|$ expanded encryption

To use BiPSA as a cipher system it would be necessary to find a pair of square matrices (Ω_e, Ω_d) such that for every ordered list of *n* integers, P, (representing plaintext), it would hold that:

$$P*\Omega_e= C \text{ and } C*\Omega_d= P$$

and further it would be required that it would be intractable to deduce one matrix from the other.

One could prepare *k* BiPSA vectors, and multiply each of them by the **W** matrix. By formal analogy to regular matrix operations, this will define <u>matrix-matrix multiplication:</u>

$\Pi * \Omega = \Phi$ {a k*m matrix, where Π is a k*n matrix – the k BiPSA vectors, and Ω is the resultant *n*m* matrix}

We can identify any element of Φ as:

$$\varphi_{ij}= [b_{i1}, b_{i2}, ...b_{in}]*[w_{1j}, w_{2j}, ...w_{nj}]$$

Assuming a square BiPSA matrix, A, the above defined matrix multiplication would lead to a power definition:

$$C = A^{k}= A*A*A....A \text{ (k times)}.$$

And from this, one would define: $A = C^{1/k}$ And a corresponding logarithm:

$$k = \log_{A}C$$

One could also define the multiplication of a scalar (t) by a matrix, corresponding to nominal matrix algebra: $C = t * A$ is defined such that: $C_{ij} = t * a_{ij}$ where a_{ij} is the element of

A in the i-th row and the j-th column. And also the addition or subtraction of two BiPSA matrices of the same dimensions (same number of rows and columns) as: D = A + C

Where A, C, and D are all BiPSA matrices of n rows, and k columns, and: $d_{ij} = a_{ij} + b_{ij}$ for i=1,2,...n and j=1,2,...k And from there one can write BiPSA polynomial equations:

$$t_n * A^n + t_{n-1} * A^{n-1} + \dots\, t_1 * A^1 = 0$$

It is easy to formulate such an equation with a given a square matrix A and n scalars t_1, t_2, t_3, ...$t_n$with **0** representing a square matrix of size A where all elements are zero. It appears quite daunting to solve the same, or even prove the existence of a solution.

4. 0 Applications

We shall focus on applications constructed from the BiPSA unit integrator. The integrator is essentially a hashing algorithm that maps any number of integers into a single integer. It is constructed in a way designed to 'fairly represent' the input integers. The term 'fairly represent' is to be understood within the context of a binary question, where the sign of the integer reflects the binary answer, and the absolute value of each integer represents the measure of certainty in the signed answer. Say then that an integer -7 indicates the binary answer of minus, with a measure of certainty greater than is exhibited by the integer -5. Accordingly, the pair of integers +M and -M will be combined to a zero -- an inability to respond to the binary question. And the pair (+4,-3) will be naturally integrated to a positive answer. The question of greater interest arises with respect to any arbitrary set of integers -- how to integrate them to a 'fair answer'. So presented, one naturally thinks of the arithmetic mean as an age old algorithm to integrate such 'binary votes with varying degrees of certainty'. Alas, the arithmetic means has numerous weaknesses when it comes to the described environment (a binary question), and the BiPSA algorithm is designed as a replacement thereto.

This property of BiPSA suggests the original application for which it was developed: discriminant analysis. And since any issue of uncertainty can be expressed as a cascade of binary questions this application is quite wide reaching. The integration property of BiPSA also immediately suggests cryptographic hashing application, and its matrix operation suggests an asymmetric cipher.
These two applications: (i) discriminant analysis and (ii) cryptographic primitives, will be briefly discussed below

4.1 Discriminant Analysis BiPSA Applications

We envision a binary question presented to *n* respondents who respond with an answer in the range {-N:+N}, where the sign of their response-integer captures their binary answer, and its absolute value captures the certainty with which the respondent gives his/her/its answer. If the n respondents are of equal trustworthiness, then their opinion (their response integers) are simply BiPSA processed, and the resulting integer reflects the wisdom of the community of the respondents as one.

If the respondents are of differing degree of trustworthiness then one constructs a **W** vector that reflects the rank-order trustworthiness of the respondents. The vector multiplication of the BiPSA vector and the **W** vector (the weight vector) will reflect the wisdom of the community of the respondents given their degree of trustworthiness.

Since the result of the BiPSA operation or the BiPSA vector multiplication is an integer in the same range of {-N:+N}, it is possible to construct a maze of BiPSA unit integrators to reflect any feedback from related previous binary questions, and increase the credibility of the overall BiPSA integration. The community of respondents may be divided to categories, the BiPSA integrated result of each category may also be computed, as well as other part way integrated results, along with the final integration.

Illustration: Alice, Bob, Carla, and David respond to the same binary question: Alice votes +2, Bob votes -1, Carla votes +1, and David votes -3. If the four respondents are regarded of equal weight and trustworthiness, then the BiPSA integrated result will be: [2,-1,+1,-3]=-1 namely, the community of respondents voted for the negative option at low certainty. If, on the other hand the trustworthiness of the respondents is: Alice=2, Bob=1, Carla=3, David=1, then the vector multiplication will determine the community vote: : [2,-1,+1,-3][2,1,3,1]=1

4.2 BiPSA Cryptographic Primitives

Representing a set of *r* characters (say the standard ASCII table for printable characters) with *r* integers is straightforward. Accordingly any plaintext P can b block chopped into blocks of *n* integers, which will be deemed BiPSA sets, and then BiPSA operated for a hash operation of *n:1* ratio. The result might be helpful for error correction. Alternatively the plaintext BiPSA entities may be vector multiplied with a secret **W** vector, to again hash at a *n:1* ratio, this time based on the secret **W** key. One could replace the **W** with a Ω matrix of k columns as a secret key and hash the plaintext blocks at a ratio of *n:k.*

Using a square Ω matrix, one would encrypt the plaintext, P to a same size ciphertext C. Alas, the corresponding decrypting matrix is not an obvious deduction. The author is preparing a report on a method to identify corresponding pairs of matrices $\Omega_{encryption}$ and $\Omega_{decryption}$ that would be hard to deduce one from the other and thereby serve as a basis for an asymmetric cipher.

Illustration: Suppose the plaintext is comprised just from the 26 letters of the alphabet. These letters can be mapped to 26 integers from -13 to +13. And hence any plaintext constructed from these letters will be a string of {-13:+13} integers. For illustration purposes we decide on chopping the plaintext to blocks of 3 letters each. We now select a random 3x3 integer matrix, say:

$$\begin{matrix} 4 & 3 & 2 \\ 1 & 1 & -1 \\ -2 & 1 & -3 \end{matrix}$$

And use it to encrypt each block in turn. If a block looks like [11,-3,7], then we compute the ciphertext as: [11,-3,7][4,1,-2], [11,-3,7][3,1,1,], [11,3,7][2,-1,-3]

5.0 Outlook and Summary

In this noisy information age, the daunting challenge of science in general and computer science in particular is to fairly and exhaustively draw conclusions from this data tsunami. BiPSA is a carefully constructed means to condense, integrate and summarize any size of original data to any desired size of summary and integration. By this attribute alone the BiPSA effort commands interest and attraction for many a disciplines. And since the science of cryptography is the opposite: means to obscure, cloud, and prevent an adversary from reading our data -- the BiPSA well handled data loss is a promising framework for many a cryptographic primitives

Reference:

- Goldreich-01: Goldreich, 2001 "Foundations of Cryptography" Cambridge University Press
- Leonard M. Blumenthal, "Addition," in AccessScience, ©McGraw-Hill Companies, 2008, http://www.accessscience.com.ezproxy.umuc.edu Samid-10: Binomial Sieve Series -- a Prospective Cryptographic Tool Cryptology ePrint Report 2010/421
- McLachlan-04: Geoffrey J. McLachlan "Discriminant Analysis and Statistical Pattern Recognition" (Wiley Series in Probability and Statistics) 2004
- Menezes-97: Menezes, Oorschot, Vanstone "Handbook of Applied Cryptography" CRC Press, 1997
- Samid-02 " At-Will Intractability Up to Plaintext Equivocation Achieved via a Cryptographic Key Made As Small, or As Large As Desired - Without Computational Penalty " 2002 International Workshop on CRYPTOLOGY AND NETWORK SECURITY San Francisco, California, USA September 26 -- 28, 2002
- Samid-10: Binomial Sieve Series -- a Prospective Cryptographic Tool Cryptology ePrint Report 2010/421