RWR Emitter Identification

- **Arthur Schwarz**
  - slipbits@yahoo.com
- **March 7, 2019**
- **14:00 – 15:00 EST**
  - 19:00 – 20:00 UTC

ASSOCIATION
OF OLD CROWS

**Stephen "Muddy" Watters,
LtCol USMC (Ret)**

AOC President

ASSOCIATION
OF OLD CROWS

AOC Virtual Series made possible by:

Gold Sponsor

ROHDE&SCHWARZ

Interested in Sponsoring the AOC Virtual Series? Click here

Presenter:

Arthur Schwarz
slipbits@yahoo.com

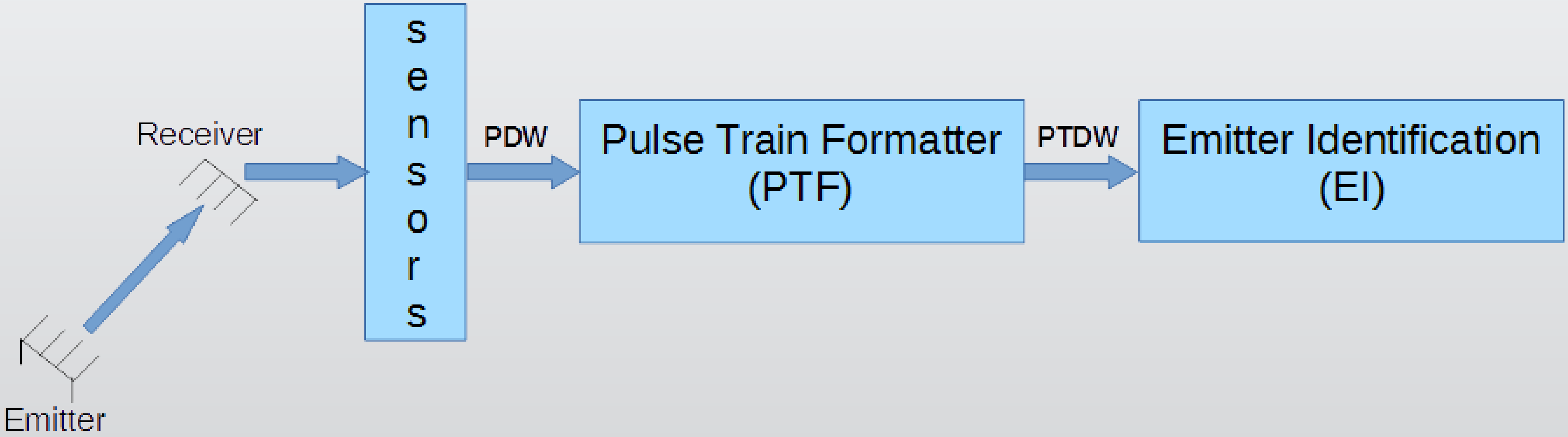
RWR Emitter Identification

- **Arthur Schwarz**
  - slipbits@yahoo.com
- **March 7, 2019**
- **14:00 – 15:00 EST**
  - 19:00 – 20:00 UTC

ASSOCIATION
OF OLD CROWS

What is a Radar Warning Receiver

- **A conceptual view of an RWR**



Receiver

Emitter

sensors

PDW

Pulse Train Formatter (PTF)

PTDW

Emitter Identification (EI)

We are interested in optimizing the EI

Goals

- Ranking identification algorithms
- Provide formulæ for algorithm analysis
  - Provide a basis for comparing algorithms
  - Provide a basis for tradeoff analysis
- Using **all** emitters during a time slice
  - Emitters for all theaters
- Increase time/space scalability

Assumptions

- Single processor/single core
- Performance is given as Big ‘O’, O(N)
  - O(N) : work = c N + k
  - Worst case analysis is done
- Total of 4096 (2^{12}) virtual emitters
  - Each mode of a multi-mode emitter becomes a virtual emitter
  - 4096 virtual emitters can represent < 4096 real emitters
- Data is aligned on a 16-bit boundary
- The PTDW is the sole source of input emitter data
  - Types of PTDW Data
    - Discrete Data, e.g., polarization (1 byte)
    - Ranged Data, e.g., frequency (4 bytes)

Assumptions

- Actual statistics are not known
- Each emitter definition is 27 bytes
  - Ranged Data, 8 bytes ([low, high])
    - Frequency (*f*) ,
    - pulse recurrence interval (PRI)
    - pulse width (PW)
  - Discrete Data, (1 byte)
    - Polarization
    - D1
    - D2
  - Call it 28 bytes

Classical Algorithm
Linear Emitter Search

- Emitter space is 2^{12} emitters times 28 bytes = 114,688 bytes
- Algorithm

```
for (all input PTDWs)
    if (func(PTDW) == true) do something }
func(PTDW) {
  for (all emitters) {
     if (Polarization == input) then
        if (D1 == input) then
          if (D2 == input) then
            if ($f_{low}$ <= input <= $f_{high}$) then
               if ($PRI_{low}$ <= input <= $PRI_{high}$) then
                 if ($PW_{low}$ <= input <= $PW_{high}$) then
                   return true; }
  return false;}
```

Classical Analysis

- Work done definition
  - Function call overhead can be ignored
    - Either copy or inline the function
  - Work for each discrete check is ‘1’
    if (discrete == true)
  - Work for each range check is ‘2’
    if (range$_{low}$ <= input) &&
    if (input <= range$_{high}$)
  - Work = N * (2 * #range + #discrete)

Classical Work Analysis
Linear Emitter Search

- Work = 4096 * (3 * 2 + 3) = 36,864 worst case

best	expected	worst	space	work
O(1) 1	O(N/2) 2048	O(N) 4096	114,688 bytes	36,864

Linear Emitter Search Optimization

- Four possibilities for optimization are:
    - Include only Theater Emitter Data
    - Threat Precedence
    - Work Reduction
    - Hybrid Search
- All optimizations use a version of the Classical Algorithm
- All Classical Algorithms used the same algorithm
- Additional optimizations do not add a benefit

Classical Algorithm Theater Emitter Optimization

- Restrict emitter data to expected Theater emitters
  - Assume 1024 emitters / theater
  - Work reduction is 75%

best	expected	worst	space	work
O(1) 1	O(N/2) 512	O(N) 1024	28,872 bytes	9,216

- Requires knowledge off Theater emitters
- Requires operational updates for each Theater

Classical Algorithm Threat Precedence

- Order the emitter table by descending threats
  - Imminent death first, cell phone last
- Assume 2^9 threats and $2^{12} - 2^9$ non-threats
  - Threat emitter identification probe count

best	expected	worst
O(1) 1	O(K/2) 256	O(K) 512

  - Non-threat emitter identification probe count

best	expected	worst
O(1) 513	O(N/2) 2304	O(N) 4096

Classical Algorithm Work Reduction

- Reorder the ‘if’ statements
  - ‘if’ statements are chosen by the size of associated emitter populations
    - Precedence is given to the most uniform population
    - Prefer discrete checks over range checks
- For example, if polarization has 8 states and each state contains 1/8 of the emitters then
  - Checking for polarization first eliminates 3,584 emitters with one check
- Work = 3,584 * (1) + 512 * (2*3 + 3) = 8,192
  - Worst case analysis

Classical Algorithm Work Reduction

- Work

best	expected	worst	space	work
O(1) 1	O(N/2) 2048	O(N) 4096	114,688 bytes	8,192

- No change in the number of probes or space
- Decrease in work / emitter
  - Work reduction is 78%
- All theater emitter data is used

Semi-Classical Algorithm Hybrid

- Suppose the discretes are packaged into a single computer word
  - Signature = polarization | D1 | D2
- Suppose there are 256 legal signatures
  - $2^{12} / 2^8 = 16$ emitters / signature
- Construct a signature for each emitter
  - Group emitters with the same signature into a list
  - Factor out the discretes from the emitter data base
- Do a binary search on the signature list and a linear search on the emitter list

Semi-Classical Algorithm

Hybrid Algorithm

- Algorithm

```
for (all PTDWS) {
  construct a signature(input PTDW)
  do a binary search of the signature list
  if (func(PTDW,list) == true) do something }
 func(PTDW, emitter list) {
  for (all emitters) {
      if ($f_{low}$ <= input <= $f_{high}$) then
          if ($PRI_{low}$ <= input <= $PRI_{high}$) then
              if ($PW_{low}$ <= input <= $PW_{high}$) then
                  return true; }
  return false;}
```

Semi-Classical Algorithm Hybrid

- Space
  - Discrete data is factored out of emitter definitions
  - Emitter data base dominates signature data
    - Total size = signature size + range data size
- Work = 1.5 * (8 * (1)) + 16 * (6) = 108
  - Work reduction is 99.7%
- Cost (S= #signatures, K = #emitters in list)

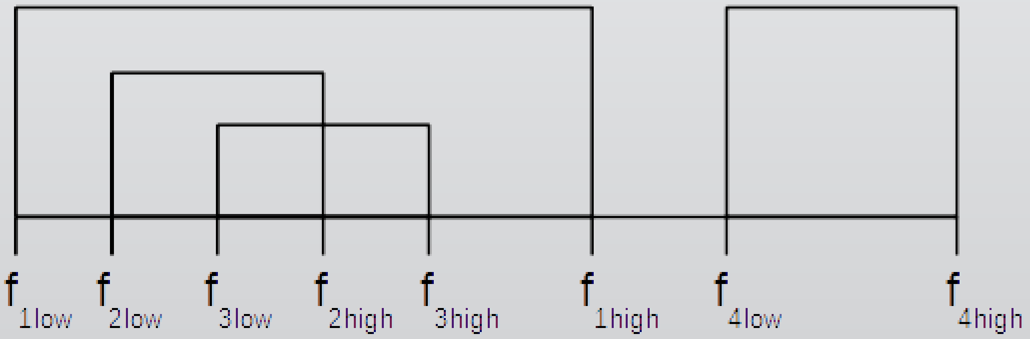
best	expected	worst	space	work
O(1) 9	$O(\log_2 S) + K/2$ 16	$O(\log_2 S) + K$ 24	99,328 bytes	108

Neo-Classical Algorithm Hybrid

- In the Hybrid Algorithm a linear search is done on range data
  - A binary search is done with the discrete signature
  - A linear search is done on the emitter list
    - Range data checks are the most expensive
- It is possible to convert emitter range data to a discrete number
  - Making it possible to construct a signature containing all of the PTDW data

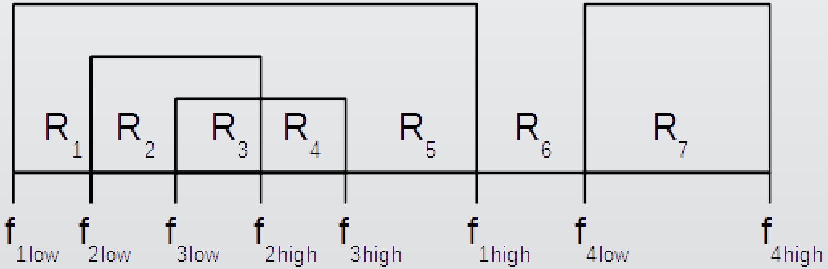
Neo-Classical Algorithm Range Decimation

- Suppose we have four frequency ranges
  - f_1, f_2, f_3, f_4
- And
  - f_2 and f_3 are wholly contained in f_1,
  - f_2 intersects f_3, and
  - f_4 is disjoint
- Graphically



Neo-Classical Algorithm Range Decimation

- This decomposes into the following regions



- The regions are disjoint
    - An emitter frequency range can be in one or more ranges
    - An emitter frequency boundary, f_{low} and f_{high}, must be on a range boundary

Neo-Classical Algorithm
Range Decimation

- Range $R_{(i)high} \approx R_{(i+1)low}$
  - Sequential ranges are disjoint
    - $[R_{low}, R_{high})_{i}, < [R_{low}, R_{high})_{i(i+1)}$
- A sorted range table can be constructed
  - table = $\{R_{(1)low}, R_{(2)low} \ldots R_{(k)high}\}$
- A Range Decimation Table is required for *f*, PRI, PW
- For an input value, a binary table search will yield the decimated range containing the value
  - If the range is R_i then the decimated number is *i*

Neo-Classical Algorithm Signature Construction

- Signature creation
  - A discrete enumeration field requires $\log_2 K + 1$ bits on the number of enumerations, e.g.,
    - If there are 5 polarizations then 3 bits must be used
  - Reserve enough bits for the size of each Range Table
    - 11 bits are needed for 1024 ranges
- Pack the result
- Sort the signatures

Neo-Classical Algorithm Signature Algorithm

- **Note:** virtual emitters are created for each range containing the emitter
  - This increases the size of the signature table
- The discrete data and decimated values are packed into the signature
- Algorithm

```
for (all PTDW) {
  signature = polarization | D1 | D2
  for (all range fields in PTDW) {
    signature |= get(PTDW.range, range table)}
  emitter = BinarySearch(signature) }
```

Neo-Classical Algorithm Costs

- Assume
  - *f* has 8192 (2^{13}) decimated ranges
  - PW, PRI each have 512 (2^9) decimated ranges
  - Polarization, D1 and D2 each have 8 values
    - Requires 3 bits for each item
- Signature size is 5 bytes (40 bits = 13 + 9 + 9 + 3*3)
  - Call it 6 bytes
- All fields are factored out of the emitter data base, leaving it empty

Neo-Classical Algorithm Signature Search

- Assume
  - Number Signatures = 8,192
  - Signature Space = 65,536 bytes = 6 * 8,192 bytes
  - Decimated ranges = 36,864 bytes
- Work = 1.5 * (13 + 9 + 9 + 13) = 66
  - Work Reduction is 99.8%
- Work

best	expected	worst	space	work
O(1) 4	$O(\log_2 S)$ 44	$O(\log_2 S)$ 44	102,400 bytes	66

Neo-Classical Algorithm Signature List Analysis

- Many emitters can share the same Region, R_i.
  - If two or more emitters have the same signatures then they are ambiguous
- The signatures can be analyzed for unneeded parameters.
  - The PTDW states what is required to generate a signal
  - This is not the same as the requirements for a search
  - It is possible that the signature over-represents search requirements

Summary Scalability

- Assume 4 times the number of emitters

Algorithm	expected	worst	Space	work
Classical	8,192	16,384	458,752	147,456
Theater	2,048	4,096	114,688	38,864
Work	8,192	16,384	458,752	32,768
Precedence			458,752	147,456
Threat	1,024	2,048		
Non-Threat	8,704	16,384		
Hybrid	40	72	396,288	396
Neo-Classical	46	46	233,472	69

Summary Comparison

- T = number threats
  N = number emitters
  S = number signatures

f = *f* table size
PRI = PRI table size
PW = PW table size

Algorithm	expected	worst	space	Work
Classical	N/2	N	28 N	9 N
Theater	N/2	N	28 N	9 N
Work	N/2	N	28 N	(N - K) + 9 K
Precedence **Threat** **Non-Threat**	T/2 (N+T)/2	T N	28 N	9 T 9 N
Hybrid	$(\log_2 S + N/S)/2$	$\log_2 S + N/S$	4 S + 24 N	$1.5 \quad \log_2 S + 6\ N/S$
Neo-Classical	$\log_2 S$	$\log_2 S$	6 S + 4 (*f* + PRI + PWI)	$1.5 (\log_2 f + \log_2 PRI + \log_2 PW + \log_2 S)$

Summary Comparison

Algorithm	Probe Count	Work	Scalability	Rating
Classical	high	high	poor	4:poor
Theater	poor	poor	poor	4:Poor
Work	high	moderate	poor	3:good
Precedence	high	poor	poor	4:Poor
Hybrid	low	low	moderate	2:better
Neo-Classical	very low	very low	good	1:best

Summary

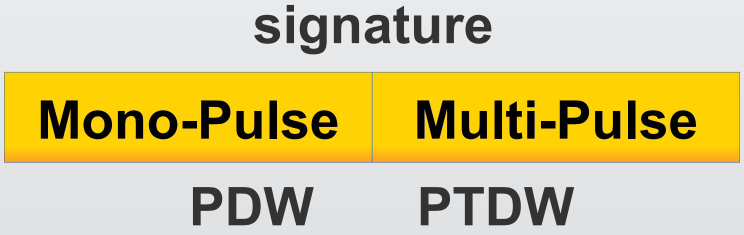
- Product Related Recommendations
    - Least impact: Work Reduction Algorithm
    - Moderate impact: Hybrid Algorithm
    - Most impact: Signature Algorithm

The Real World

- Some things to keep in mind
  - Combinatorics
    - Each mode of a multi-mode emitter creates a virtual emitter
  - Evaluate algorithms using your statistics
  - CPU’s with large primary and secondary cache alter the work effort but don’t alter the result
  - Multi-core CPU’s alter the work effort but don’t alter the result
- Summary: Using real world figures and/or implementations will alter the details but what’s good is good and what’s bad is bad.

Speculation
Front-end Algorithm Use

- A Signature is composed of two parts



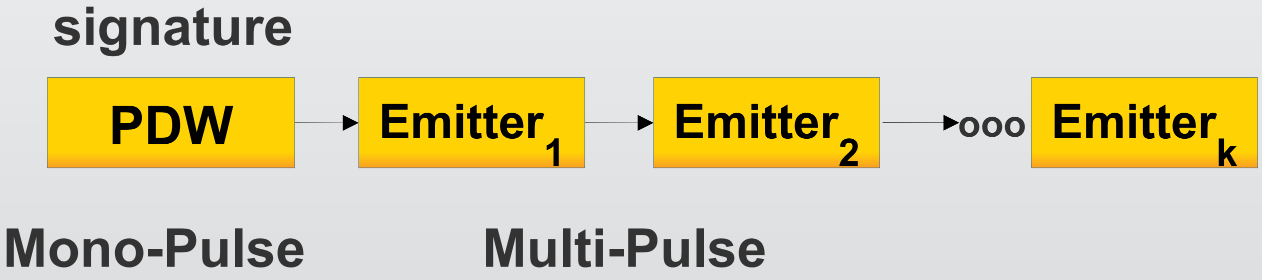
signature

Mono-Pulse	**Multi-Pulse**
PDW	**PTDW**

- Removing dynamic data, e.g., PRI from the PTDW creates an ambiguous signature
  - Multiple emitters with the same signature
  - Multi-pulse discrimination uses multiple mono-pulse data

Speculation
Front-end Algorithm Use

- The resulting structure looks like.



signature

PDW → **Emitter$_1$** → **Emitter$_2$** → ooo **Emitter$_k$**

Mono-Pulse **Multi-Pulse**

- At the cost of a linear search over a reduced Emitter space we get emitter identification in the front-end
- Use of geolocation, speed and distance is an architectural issue