Trust based Evaluation System using Signal Noise Detection for MANET and Noise Reduction by Comparative Analysis between Alpha Beta Filter and Kalman Filter

Jayanta Das
SSVASM
7/2A PWD Road
Kolkata – 35, WB, India

Abhijit Das
RCCIIT
Canal South Road, Beliaghata
Kolkata – 15, WB, India

ABSTRACT
Security and trust are two inevitable concept for secure MANET. There are various systems used for ensuring security and trust in case of MANET. These systems has several advantages as well as several disadvantages in terms high communication and computation overhead. In this proposed trust based system, trust is evaluated on the basis of detection of signal noise and after that reduction of noise as much as possible with the help of Alpha Beta Filter as well as Kalman filter once the signal is flowing from one node to another node. In this paper, it is also able to show that using Kalman filter is more advantageous than alpha beta filter for reducing the error due to noise.

General Terms
Mobile Communication, Security, Algorithms.

Keywords
Alpha Beta Filter, MANET, Kalman Filter, Security, signal Noise, Trust

1. INTRODUCTION
Trust is one of the prime factor for decision-making processes of any network for which uncertainty plays very crucial role. If it can be predicted regarding any faulty behavior in advance for any network, any network becomes flawless. Out of these faulty behaviors, noise is one of them. Noise is detected with the help of common term known as "signal-noise ratio".

2. RELATED WORK
In [1], nodes are created and deployed in the wireless sensor network. These nodes are labeled as CH or Cluster Head, CM or Cluster Mode and Bs or Base Station. Each CH includes several CMs. A lightweight trust scheme is evaluated between CMs or between CHs. Within each cluster, CH evaluates indirect trust for its CM. So, CM does not require to maintain feedback from other CMs which reduces the communication overhead. According to dependability-enhanced trust evaluating approach, CHs take the responsibility of large amount of data forwarding and communication tasks which reduce network consumption by preventing improper (malicious, selfish and faulty) CHs. In [2], wormhole attack affects severely the routing and byzantine attack weakens the routing services. To overcome these attacks, the proposed trust based approach evaluates the Observed Trust Value or OTV and Advertised Trust Value or ATV. The Observed Trust Value indicates to the root trust calculated by node itself on the basis of ROUTE_ACKNOWLEDGEMENT or R_ACK information. The Advertised Trust Value is advertised by a node which is located at downstream of current node. Route Selection Value or RSV is calculated with the Observed Trust Value and the Advertised Trust Value. Route Selection Value is acted as a parameter in choosing one of the multiple paths leading towards destination. Whatever the path chosen by Route Selection Value is the trusted and shortest path. In [3], according to Seniority Based or SB trust model, trust management and maintenance are distributed in both space(k) and time(T) domains. Thus Seniority Based model depicts a seniors-securing scheme to node authentication in MANET. In other words, the time varying feature of a trust relationship, while k indicates the number of senior nodes required to work as 'Certificate Authority' or CA. In the network, there are several groups. Again each group consists of several nodes. The leader of the group acts as a 'Certificate Authority' or CA, which issues group membership certificates. CAs certify that the public key in the certificate belongs to a group member. An entity is trusted if any k trusted available senior entities claim so within a specific time period say, T. Once a node is trusted by its groups, it is universally accepted as a trusted node. Otherwise, if the seniors distrust a node, this node is treated as untrustworthy in the entire network.

3. PROPOSED SYSTEM
According to the proposed system, at first it has to detect noise of signal coming from any particular node. For this purpose, it has to calculate SNR or Signal to Noise Ratio. If this value is greater than 1, it can be easily determined that the node, from which the signal is coming, is trustworthy. Otherwise, if this Signal to Noise Ratio is less than 1, it can be concluded that the node is untrustworthy. At that time, two specific filters known as Kalman filter and alpha beta filter are used to minimize the effect of noise as much as possible so that the Signal to Noise Ratio becomes greater than 1 and the node obtains trustworthy status from previous untrustworthy status.

4. BACKGROUND STUDY

4.1 Noise
Noise [4] can be generally described as, which, when interpreted by receiving node, delivers absurd information without any interest to the receiver. Noise gets added with signal at the time of signal transmission and produces inferior quality signal. Background noise is always present in spite of absence of useful signal. Sources of background noise are:-

- Thermal noise
- Intrinsic noise of the electronic devices, for example, shot nose

- Atmospheric disturbances
- Electromagnetic interference

Lightning and rain attenuation are two sources of atmospheric noise. Electromagnetic interference is caused by discharges in commutator motors and spark plugs of vehicles. Sources of thermal and shot noise are present within the telecommunication equipment.

4.2 Signal to Noise Ratio

The quality of a signal is calculated by its level with respect to the level of noise, which is included into the signal. It is termed as the ratio of signal power(P_s) to noise power(P_n) and is known as the Signal-to-Noise Ratio(SNR). It is represented as following:-

$$S / N = Ps / P_n$$

4.3 Alpha Beta Filter

Alpha-Beta (α-β) [7] filters were employed to reduce the mean square error in estimating position and velocity. According to assumption of this filter, the velocity remains more or less constant over the small time period or the sampling rate. Thus, Alpha Beta filters have very limited capacity to track accelerating (changing direction) targets. According to this filter technique, the system is sufficiently approximated by a model having two internal states. The two states are position *X* and velocity *V*.

Because the previous *x* estimate was low, the previous *v* was low, or some combination of the two, so as a result of this, it can be assumed that the residual *r* is positive. The alpha beta filter takes selected *alpha* and *beta* constants (from which the filter gets its name), uses *alpha* times the deviation *r* to correct the position estimate, and uses *beta* times the deviation *r* to correct the velocity estimate. An extra ΔT factor conventionally serves to normalize magnitudes of the multipliers.

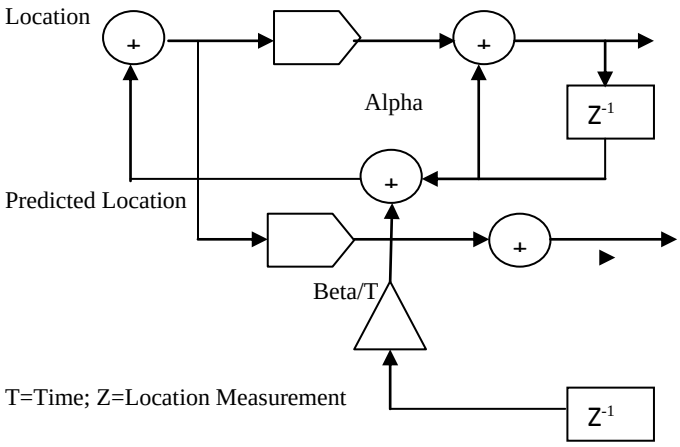

Figure1: Alpha Beta Filter

4.4 Kalman Filter

- A Kalman Filter is an optimal data processing algorithm.
- The Kalman Filter incorporates all information that can be provided to it. It estimates the current value of the variables of interest after processing all available information regardless their precision.
- For Provides current parameters' estimation using current measurements and previous parameter estimates
- Should provide a close to optimal estimate if the physical situation is matched the models used in filter

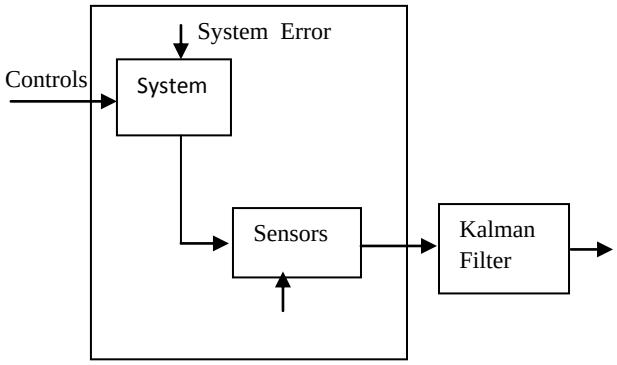

Figure2: Kalman Filter

5. SYSTEM DESCRIPTION

Step 1: A structure 'AlphaBeta' is declared. Under this structure, the variables 'alpha'(i.e. alpha value which effects position x), 'beta'(i.e. beta value which effects velocity v), 'xk_1'(current x-estimate value) and 'vk_1'(current v-estimate value) are declared.

Step 2: After that current system state(i.e. position) 'xk', derivative of system state(i.e. velocity) 'vk' and residual error 'rk' are declared.

Step 3: Update position(estimated) state 'x' from the system(i.e. position = position + velocity(last).dt with following way,

$$xk = xk\_1 + dt \times vk\_1$$

We know, velocity= d/dt(position); So position= dt × velocity

velocity is derivative of position with respect of time that means dt

Step 4: Update(estimated) velocity in following way,

$$vk = vk\_1$$

Step 5: Calculate residual error as difference of measured value and estimated value of position in following way,

$$rk = x\_measured - xk$$

Step 6: Update the estimates given the residual error in following way,

$$xk = xk + alpha \times rk$$

$$vk = vk + beta/dt \times rk$$

Step 7: Now all current values become old values for next time in following way,

$$vk\_1 = vk$$

$$xk\_1 = xk$$

where, vk and xk be current values of velocity and position respectively. vk_1 and xk_1 be old values of velocity and position respectively.

Step 8: Now one function 'frand()' is defined under which system random value is measured.

Step 9: Enter the value of signal and noise.

Step 9A: If signal value is greater than noise value, that means ratio between signal and noise is greater than 1, it can be concluded that node or system is trustworthy

Step 9B: Otherwise, it can be concluded that node or system is untrustworthy., that means system has error due to noise. Now go to Step 10 to reduce error using Alpha Beta filter by calling function 'filterAlphaBeta()' and then go to Step 18 to reduce error using Kalman filter by calling function 'Kalman()'.

Step 10: Function 'filterAlphaBeta()' is defined. Under this function, the following variables are declared:

- 't' as time,
- 'x' and 'y' as ideal values under 'x' and 'y' co-ordinate
- "xnoise' and 'ynoise' as inserted noise values under 'x' and 'y' co-ordinate(this noise is inserted into our system).Both 'xnoise' and 'ynoise' are initialized as 0
- 'merror' as error due to difference between measured value and ideal value
- 'ab_error' as error due to difference between value using Alpha Beta filter and ideal value

Step 11: Initialize Alpha Beta filter values with respect of 'x' and 'y' position.

Step 12: Indicate system's true position, as x = cos(t) and y = sin(t)

Step 13: Update simulated noise in following way,

xnoise = xnoise + frand()× ratio × 0.01 and ynoise = ynoise + frand()× ratio × 0.01

Here 'ratio' indicates ratio between signal value and noise value

Step 14: Calculate measured position including some noise

xm = x + xnoise and ym = y + ynoise

Step 15: Adjust system's filtered position by removing noise.

Step 16: Now print following values:

- Ideal position
- Measured position as fabs(x - xm) and fabs(y – ym)
- Alpha Beta position as fabs(x – ab_x.xk_1) + fabs(y – ab_y.xk_1)

where fabs (a) be absolute value of 'a'

Step 17: Update the errors as m_error = m_error + fabs(x – xm) + fabs(y – ym)

ab_error = ab_error + fabs(x – ab_x.xk_1) + fabs(y – ab_y.xk_1)

Step 18: Calculate reduction in error using Alpha Beta filter as 100 – (int) ((ab_error / m_error) × 100)

Step 19: Function 'Kalman ()' is defined. Under this function, the following variables are declared:

- 'x_est_last' as last system's estimated value and initialized as 0
- 'p_last' as system's last predicted value and initialized as 0
- 'Q' and 'R' as two different noise values and initialized as 0.022 and 0.617 respectively
- 'K' as Kalman gain
- 'p' as current predicted value
- 'p_temp' as temporary predicted value
- 'x_temp_est' as system's temporary estimated value
- 'x_est' as system's current estimated value
- 'z_measured' as measured noisy value
- 'z_real' as measured ideal value and initialized as 0.5

Step 20: Calculate system's last estimated value as

x_est_last = z_real + frand() ×ratio ƒ 0.09

Here, 'ratio' indicates ratio between signal value and noise value.

Step 21: Initialize error due to ideal situation as sum_error_measure = 0 and error due to using Kalman filter as sum_error_kalman = 0

Step 22: Predict some measurement for some no. of loops (say 50) in following way

x_temp_est = x_est_last and p_temp = p_last + Q

Step 23: Calculate Kalman gain as

K = p_temp ƒ (1.0 / (p_temp + R))

Step 24: Calculate the real measurement plus noise

as z_measured = z_real + frand() × ratio

Step 25: Update the system's estimated value and predicted value as

x_est = x_temp_est + K × (z_measured – x_temp_est)

p = (1 – K) × p_temp

Step 26: Now print following values:

- Ideal position
- Measured position as fabs(z_real – z_measured)
- Kalman position as fabs(z_real – x_est)

Step 27: Update error values

sum_error_kalman = sum_error_kalman + fabs(z_real – x_est)

sum_error_measure = sum_error_measure + fabs(z_real - z_measured)

Step 28: Update system's last values

p_last = p and x_est_last = x_est

Step 29: Calculate reduction in error using Kalman filter as 100 – (int)((sum_error_kalman / sum_error_measure) × 100)

6. PSEUDOCODE

```
typedef struct {
   float alpha; //alpha value (effects x, eg pos)
   float beta; //beta value (effects v, eg vel)
   float xk_1; //current x-estimate
   float vk_1; //current v-estimate
} AlphaBeta;

 void InitializeAlphaBeta(float x_measured, float alpha, float beta, AlphaBeta* pab) {
   pab->xk_1 = x_measured;
   pab->vk_1 = 0;
   pab->alpha = alpha;
   pab->beta = beta;
}

void AlphaBetaFilter(float x_measured, float dt, AlphaBeta* pab) {
   float xk_1 = pab->xk_1;
   float vk_1 = pab->vk_1;
   float alpha = pab->alpha;
   float beta = pab->beta;

   float xk; //current system state (ie: position)
   float vk; //derivative of system state (ie: velocity)
   float rk; //residual error

   //update our (estimated) state 'x' from the system (ie pos = pos + vel (last).dt)
   xk = xk_1 + dt * vk_1;
   //update (estimated) velocity
   vk = vk_1;
   //what is our residual error (mesured - estimated)
   rk = x_measured - xk;
   //update our estimates given the residual error.
   xk = xk + alpha * rk;
   vk = vk + beta/dt * rk;
   //finished!

   //now all our "currents" become our "olds" for next time
   pab->vk_1 = vk;
   pab->xk_1 = xk;
}

double frand() {
return 2*((rand()/(double)RAND_MAX) - 0.5);
}

float filterAlphaBeta(float ratio) {
   AlphaBeta ab_x;
   AlphaBeta ab_y;
   double t; //time
   double x,y; //ideal x-y coordinates
   double xm,ym; //measured x-y coordinates
   double xnoise = 0; //noise we are inserting into our system
   double ynoise = 0;
   double m_error = 0; //total error (measured vs ideal)
   double ab_error = 0; //total error (ab filter vs ideal)
#define DT 0.1
   //intialize the AB filters
   InitializeAlphaBeta(1,0.85,0.001,&ab_x); //x position
   InitializeAlphaBeta(0,1.27,0.009,&ab_y); //y position
   srand(0);

   for (t = 0; t < 4; t+=DT) {
      //our 'true' position (A circle)
      x = cos(t);
      y = sin(t);
      //update our simulated noise & drift
                  xnoise += frand()*ratio*0.01;
                  ynoise += frand()*ratio*0.01;
      //our 'measured' position (has some noise)
      xm = x + xnoise;
      ym = y + ynoise;
      //our 'filtered' position (removes some noise)
      AlphaBetaFilter(xm,DT, &ab_x);
      AlphaBetaFilter(ym,DT, &ab_y);

      //print
      printf("Ideal    position: %6.3f %6.3f\n",x,y);
      printf("Mesaured position: %6.3f %6.3f [diff:%.3f]\n",xm,ym,fabs(x-xm) + fabs(y-ym));
      printf("AlphaBeta position: %6.3f %6.3f [diff:%.3f]\n",ab_x.xk_1,ab_y.xk_1,fabs(x-ab_x.xk_1) + fabs(y-ab_y.xk_1));

      //update error sum (for statistics only)
      m_error += fabs(x-xm) + fabs(y-ym);
      ab_error += fabs(x-ab_x.xk_1) + fabs(y-ab_y.xk_1);
   }
   printf("Total error if using raw measured: %f\n",m_error);
   printf("Total error if using a-b filter:   %f\n",ab_error);
         printf("Reduction in error: %d%% \n",100-(int)((ab_error/m_error)*100));
   return 0;
}

float Kalman(float ratio) {

//initial values for the kalman filter
float x_est_last = 0;
float P_last = 0;

//the noise in the system
float Q = 0.022;
float R = 0.617;
float K;
float P;
float P_temp;
float x_temp_est;
float x_est;
float z_measured; //the 'noisy' value we measured
float z_real = 0.5; //the ideal value we wish to measure

srand(0);
//initialize with a measurement
x_est_last = z_real + frand()*ratio*0.09; //frand()*0.09;

float sum_error_kalman = 0;
float sum_error_measure = 0;

for (int i=0;i<50;i++) {
//do a prediction
x_temp_est = x_est_last;
P_temp = P_last + Q;
//calculate the Kalman gain
K = P_temp * (1.0/(P_temp + R));
//measure
z_measured = z_real + frand()*ratio;//frand()*0.09; //the real measurement plus noise
//correct
x_est = x_temp_est + K * (z_measured - x_temp_est);
```

```
P = (1- K) * P_temp;
//we have our new system

printf("Ideal    position: %6.3f \n",
printf("Mesaured position: %6.3f [diff:%.3f]\n",z_measured,fabs(z_real-z_measured)));
printf("Kalman position: %6.3f [diff:%.3f]\n",x_est,fabs(z_real - x_est));

sum_error_kalman += fabs(z_real - x_est);
sum_error_measure += fabs(z_real-z_measured);

//update our last's
P_last = P;
x_est_last = x_est;
}

printf("Total error if using raw measured: %f\n",sum_error_measure);
printf("Total error if using kalman filter: %f\n",sum_error_kalman);
printf("Reduction in error: %d%% \n",100-(int)((sum_error_kalman/sum_error_measure)*100));

return 0;
}
……..
int main() {
int signal,noise;
float ratio;
float Kalman(float );
float filterAlphaBeta(float );

printf("Enter the value of signal(in decibel)");
scanf("%d",&signal);
printf("Enter the value of noise(in decibel)");
scanf("%d",&noise);

ratio=(float)signal/(float)noise;

if(ratio>=1.00)
{
printf("The Node is Trustworthy");
}
else
{ printf("The Node is Untrustworthy\n\n");
printf("Now use Alpha Beta Filter to Reduce Error due to Noise\n\n" );
filterAlphaBeta(ratio);
printf("Now use Kalman Filter to Reduce Error due to Noise\n\n" );
Kalman(ratio);
}
return 0;
}
```

7. STRENGTH OF ALGORITHM

In the first time, it can be able to highlight signal noise as an indicator with respect of trust evaluation scheme.

- So it can be able to filter out the noise as much as possible so that a node becomes trustworthy.
- In this paper, it can be also able to highlight the comparative analysis between Kalman filter and alpha beta filter on the basis of error reduction due to signal noise.

Following are the results based on some random input.

```
Enter the value of signal(in decibel)89
Enter the value of noise(in decibel)123
The Node is Untrustworthy

Now use Kalman Filter to Reduce Error due to Noise

Mesaured position:  0.436 [diff:0.064]
Ideal    position:  0.000
Kalman   position:  0.435 [diff:0.065]
Mesaured position:  0.435 [diff:0.065]
Ideal    position:  0.000
Kalman   position:  0.435 [diff:0.065]
Mesaured position:  0.479 [diff:0.021]
Ideal    position:  0.000
Kalman   position:  0.439 [diff:0.061]
Mesaured position:  0.439 [diff:0.061]
Ideal    position:  0.000
Kalman   position:  0.439 [diff:0.061]
Mesaured position:  0.481 [diff:0.019]
Ideal    position:  0.000
Kalman   position:  0.444 [diff:0.056]
Mesaured position:  0.463 [diff:0.037]
Ideal    position:  0.000
Kalman   position:  0.447 [diff:0.053]
Mesaured position:  0.505 [diff:0.005]
```

Figure3: Input value II

```
AlphaBeta position: -0.755 -0.710 [diff:0.052]
Ideal     position: -0.705 -0.709
Mesaured  position: -0.742 -0.723 [diff:0.051]
AlphaBeta position: -0.744 -0.725 [diff:0.055]
Ideal     position: -0.683 -0.730
Mesaured  position: -0.719 -0.753 [diff:0.059]
AlphaBeta position: -0.723 -0.759 [diff:0.069]
Ideal     position: -0.661 -0.750
Mesaured  position: -0.706 -0.780 [diff:0.075]
AlphaBeta position: -0.708 -0.785 [diff:0.082]
Total error if using raw measured: 6.515393
Total error if using a-b filter:   6.055330
Reduction in error: 7.061170%
Now use Kalman Filter to Reduce Error due to Noise

Mesaured position:  0.436 [diff:0.064]
Ideal    position:  0.000
Kalman   position:  0.435 [diff:0.065]
Mesaured position:  0.435 [diff:0.065]
Ideal    position:  0.000
Kalman   position:  0.435 [diff:0.065]
Mesaured position:  0.479 [diff:0.021]
Ideal    position:  0.000
Kalman   position:  0.439 [diff:0.061]
Mesaured position:  0.439 [diff:0.061]
```

Figure4: Untrustworthy Node on Input Value II (Alpha Beta)

```
Mesaured position:  0.469 [diff:0.031]
Ideal    position:  0.062
Kalman   position:  0.500 [diff:0.000]
Mesaured position:  0.458 [diff:0.042]
Ideal    position:  0.076
Kalman   position:  0.492 [diff:0.008]
Mesaured position:  0.490 [diff:0.010]
Ideal    position:  0.097
Kalman   position:  0.492 [diff:0.008]
Mesaured position:  0.525 [diff:0.025]
Ideal    position:  0.102
Kalman   position:  0.498 [diff:0.002]
Mesaured position:  0.456 [diff:0.044]
Ideal    position:  0.115
Kalman   position:  0.491 [diff:0.009]
Mesaured position:  0.505 [diff:0.005]
Ideal    position:  0.148
Kalman   position:  0.493 [diff:0.007]
Mesaured position:  0.519 [diff:0.019]
Ideal    position:  0.153
Kalman   position:  0.498 [diff:0.002]
Total error if using raw measured:  1.547317
Total error if using kalman filter: 0.953511
Reduction in error: 38.376505%
```

Figure5: Untrustworthy Node on Input Value II (Kalman)

```
Enter the value of signal(in decibel)123
Enter the value of noise(in decibel)89
The Node is Trustworthy
```

Figure6: Trustworthy Node on Input Value I

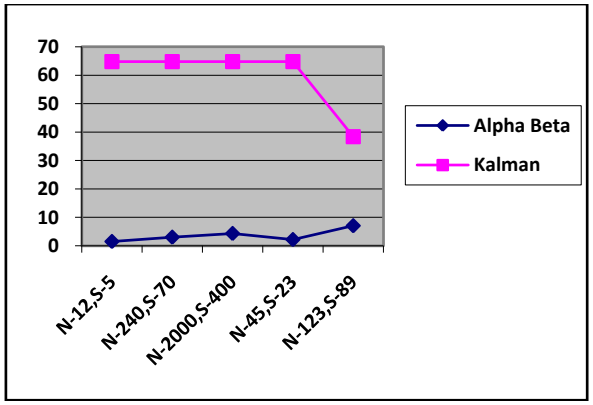

Figure7: Comparative Analysis between Alpha Beta filter and Kalman filter; N-Noise, S-Signal

Table1: Table for comparative analysis between Alpha Beta filter and Kalman filter performance

Noise value in decibel	Signal value in decibel	Error reduction by Alpha Beta filter in %	Error reduction by Kalman filter in %
12	5	1.527961	64.781996
240	70	3.042935	64.781997
2000	400	4.335037	64.781995
45	23	2.198511	64.782003
123	89	7.061170	38.376505

8. CONCLUSION

In this proposed system, problem has arisen due to performing too much computation. So computational overhead plays crucial role in this proposed system.

In future, our aim will be to try to minimize this computational load as much as possible.

9. FUTURE SCOPE

In future, the main aim should be to minimize the above massive computational overhead.

For getting better performance, the experiment should be performed regarding comparative analysis between Kalman filter and Weiner filter because the main importance of Weiner filter is to minimize amount of a noise in a signal with the help of comparing by estimating the desired noise signal.

10. REFERENCES

[1] S. Jeyantha Jafina Juliet Jacquet , M. Varghese " Role Based and Energy Efficient Trust System for Clustered Wsn." IOSR Journal of Computer Engineering, Volume 16, Issue 2, Ver. 1(Mar-Apr. 2014), pp 44-48.

[2] Abhijit Das, Soumya Sankar Basu, Atal Chaudhuri, " A Novel Security Scheme for Wireless Adhoc Network".

[3] Abhijit Das, Atiqur Rahman, Soumya Sankar Basu, Atal Chaudhuri, "Energy Aware Topology Security Scheme for Mobile Ad Hoc Network" .

[4] Jashanvir Kaur, Er. Sukhwinder Singh Sran, "SBPGP Security based Model in Large Scale Manets" International Journal of Wireless Networks and Communications, Volume 5, Number 1(2013), pp 1-10 .

[5] Eli Brookner, "Tracking and Kalman Filtering Made Easy," Wiley-Interscience, April 1998

[6] Mayiatis, D. E., "Comparison of an Alpha-Beta and Kalman Filter in Track While Scan Radars," NAVAL POSTGRADUATE SCHOOL MONTEREY CA, Diplomarbeit, 1976. http://www.stealthskater.com/ Documents/Radar_02.pdf [21 March, 2013]

[7] Wikipedia, "Alpha beta filter," 6 February 2013. [Online]. Available: http://en.wikipedia.org/wiki/Alpha_beta_filter

[8] Nidh Mittal, Janish " Performance Evaluation of AODV and DSDV under Seniority based Pretty Good Privacy Model " International Journal of Scientific and Engineering Research, Volume 4, Issue 6, June 2013, pp 943-949.

[9] Prakash C. Gupta "Data Communications " Publisher-Prentice Hall of India, Version May 1999, pp-59.

[10] John Davies " Use of Kalman filters in time and frequency analysis ." National Physical Laboratory, 1st May, 2011.

[11] Welch, G and Bishop, "An Introduction to the Kalman Filter ." http://www.cs.unc.edu/~welch/kalman/, 2001