Journal of Engineering and Applied Sciences

MEDWELL PUBLICATIONS SINCE 2000

Traffic Analysis of Anonymity Protocol using HMM-Based Trust Model

Yousif H. Sulaiman
Al-Maaref University College, Al-Ramadi, Iraq

Key words: Tor anonymity network, web safety, intruder detection system, traffic studies

Corresponding Author:
Yousif H. Sulaiman
Al-Maaref University College, Al-Ramadi, Iraq

Page No.: 2390-2397
Volume: 15, Issue 11, 2020
ISSN: 1816-949x
Journal of Engineering and Applied Sciences
Copy Right: Medwell Publications

Abstract: The implementation of a traffic attack in order to Tor network analysis is considered in this study. Tor performance and previous attacks against Tor network by other researchers presented in the references were studied. This gives us some insight into the growth in popularity of the anonymity network. Then the experimental environment is set up and the experimental procedure is built. The results of the experiment indicate the need for setting the model confidence for the analysis of anonymity protocol. The effectiveness of the suggested algorithm based trust model is demonstrated by calculating the amount of training data set required for the output of wireless access protocol proxies through Tor.

INTRODUCTION

The study describes a scenario in which a client interacts with a server through Tor. Assuming that the communication protocol used by the client can be submitted by a Hidden Markov Model (HMM), we can derive a model that is an exact submission of the basal protocol utilization the time information collected on the server side. The suggested trust model approach is applied to the attack experiment to determine the size of the data required for construct a statistically significant representative of the protocol.

Model confidence

z-test: Among several classic statistical tests, z-test is a simple but widely used statistical test. The rationale for this test is: given the random sample size n that is a sequence of random variables independently taken and equally distributed (iid) from an unheard allocation, we are going to make a solution for every value and this decision will be either correct or not. Consider the allocation of the number of errors that will be made by our arrangement system. So, far as every solution is irrespective of the others and is binary, it is sensible to expect that the casual variable X, representing the number of errors should abide by the binomial distribution B(n, p) where p-error rate. In addition, it is known that the binomial distribution B(n, p) can be approximated by a normal allocation N(μ, σ^2) with:

$$\mu = np \text{ and } \sigma^2 = np(1\text{-}p) \tag{1}$$

when n is big enough. Finally, if Y = X/n~N (p, p(1-p)/n), then, the error frequency distribution is Y = X/n~N (p, p(1-p)/n) thus:

$$\frac{\sqrt{n}(X\text{-}\mu_0)}{\sqrt{p(1\text{-}p)}} \sim N(0,1)\ .$$

Let's take sample $x_1, x_2, ..., x_n$. The null hypothesis is that the expected value x is a given value μ_X. Then, we can write test statistics like this:

$$z = \sigma_{\bar{x}} = \frac{\bar{X}\text{-}\mu_X}{\sigma_{\bar{x}}} \tag{2}$$

Where:

$$\sigma_{\bar{X}} = \frac{\sigma_X}{\sqrt{n}}$$

and σ_X dispersion X. The conversion process (2) is called standardization or normalization and the result is called the standard estimate or z-count. z-estimate determines how many standard deviations below or above the population means that the average value of the sample is under the null hypothesis. However, in most cases σ_x unknown and may be replaced by sample variance $x_1, x_2, ..., x_n$. if n big enough. Using test statistics z for a given level of significance a we compute one way or two way p-value. We reject the null hypothesis if p-value is less α and takes it another way.

Or, since, the statistics z follows the standard normal distributions, if the null hypothesis is correct, the decision to reject the null hypothesis can also be made by comparing the statistics z with a critical value without converting it to p-value.

MATERIALS AND METHODS

Hidden Markov model: The standard Hidden Markov Model (HMM) is N-Markov reverse circuit watched at discrete moments in time t = 0, 1, 2. Let us assume that S = {1, 2, ..., N} provide the area of the final state when we utilize a occasionally value S^t to indicate the condition of the HMM at the time t, $S_t = s$, denotes that the HMM is in condition of $s \in S$ in time step t. However, S_t cannot be observed directly. Instead, we see one way out. $O_t = o \in O$ where, O = {1, 2, ..., M} stands for the final kit of exits as well known so supervisions. For each state $s \in S$ two probability distributions are defined to represent the state transition and output radiation rules, respectively: Transition state theory:

$$P_s^{s'} = \Pr\{S_{t+1} = s' | S_t = s\}, \quad \forall s, s' \in S, t = 0, 1, 2, ... \tag{3}$$

and probability of observations:

$$P_s^{o} = \Pr\{O_t = o | S_t = s\}, \quad \forall s \in S, o \in O, t = 0, 1, 2, ... \tag{4}$$

As shown in Fig. 1, two HMM probability distributions generate two parallel stochastic processes[1] a process of states and a state of observations.

In this study, we consider the problems of HMM inference and a specific inference algorithm the causal splitting restoration algorithm (CSSR). This approach of the HMM[2] creates condition apparatus certain at the conversion exit, i.e., when every supervision is displayed in no more than one transition, leaving the state. In addition, the main Markov chain HMM generated using the Shalizi method is irreducible when all transition states are removed[2].

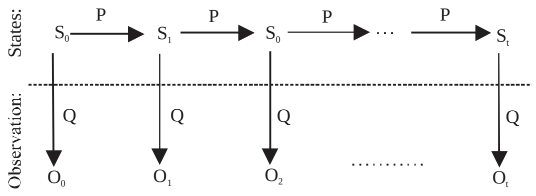

Fig. 1: The HMM process and its two stochastic processes: probability of $\{S_t\}$ and the observation process $\{O_t\}$

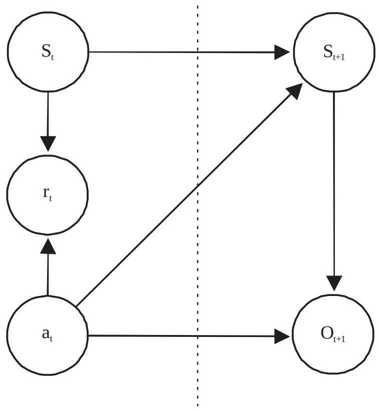

Fig. 2: An influence diagram showing the relationships

Fractionally observable Markov solution act: In the language of stochastic control, Partially Observable Markov Decision Process (POMDP) are control problems with partial observation. They usually simulate stochastic environments with hidden processes. By summarizing the Markov Decision Process (MDP) and providing greater uncertainty, the POMDP provides a more powerful formalism for modelling realistic problems, especially, for managing systems with noisy data or limited sensitivity. Formally, POMDP is given as a 6-tuple $\langle S, A, O, T, Z, r\rangle$ where:

- S = {1, 2, ..., N} final kit of conditions
- A = {1, 2, ..., M} finite course of actions
- O = {1, 2, ..., K} finite set of observations or exits.
- $T: S\times A\times S \rightarrow [0, 1]$ state-transition function. T(s, a, s') = Pr(s'|s, a) represents the probability of transition to state after taking measures a in the condition s
- $Z: A\times S\times O \rightarrow [0, 1]$ probability function of observation Z(a, s', o) = Pr(o|a, s') denotes the possibility of seeing o in the state's after adoption action a at the preceding step
  $r: S\times A7 \rightarrow R$ direct remuneration function

Figure 2 an influence diagram showing the relationships between the various elements containing POMDP which meanwhile prove the Markov property. Solid arrows represent the dependencies of existence (for example, S_{t+1} depend both on S_t and on A_t).

At any time, the system is in some condition. $s \in S$. agent accepts action $a \in A$ which immediately gives a reward r(s, a) and starts the transition to the new state $s' \in S$ in the next step with probability T(s, a, s'). Three elements S, A and T form the core-MDP and determine the dynamics of the POMDP. But unlike conventional MDP, the agent cannot observe the state of the MDP core during the decision process. Instead, he gets an observation $o' \in O$ with probability function Z(a, s'o').

Unlike standalone HMMs, POMDPs are controlled by actions selected by agents or controllers. The goal of the POMDP study is to find a sequence of actions $\{At\}_{t = 1, 2 ...}$ known as the policy that makes the system work as agents want A policy is measured by a compensation function which is a mathematical function of immediate remuneration. The goal of the agent is to optimize the compensation function.

Decentralized POMDP: When decision making becomes a collective work in which several agents need to be coordinated without effective communication and even unclear about their own local situation, the decentralized controlled Markov processes with partial observations (DEC-POMDP) is the main tool used in decision theory to solve this problem[3]. As an extension of POMDP to the case of several agents, DEC-POMDP is a more general and more powerful modelling tool. However, the DEC-POMDP solution usually leads to excessive computational overhead.

DEC-POMDP can be formally defined by a cortege $\{S, K, A_l, ..., A_K, O_l, ..., OK, R, P, Q\}$ where:

- S-finite set of states
- K-number of agents
- $A^{(k)}$, $1 \leq k \leq K$-agent's action space k
- $O^{(k)}$, $1 \leq k \leq K$-agent's observation space k
- $R: S \times A^{(1)} \times, ..., \times A^{(K)} \rightarrow R$–direct reward function
- $P: S \times A^{(1)} \times, ..., \times A^{(K)} \times S \rightarrow [0, 1]$ state stochastic transition function

Let us assume that $\bar{a} = [a_1, a_2, ..., a_K]$ where, $a_k \in A^{(K)}$, P(s, a. s') represents the probability of transition from the state to the state s'.

- $Q: A^{(1)} \times, ..., A^{(K)} \times S \times O^{(1)} \times, ..., O^{(K)} \rightarrow [0, 1]$ state stochastic transition function

Let us assume that $\bar{o} = [o_1, o_2, ..., o_K]$ where $o_k \in O^{(K)}$, $Q(\bar{a}, s', \bar{o})$ represents the probability of obtaining sequence of observations $\bar{o}$ in the state s' after the agent k, $1 \leq k \leq K$ takes action a_k at the preceding step.

From the definition, we can see that each agent needs a local policy. But joint actions affect both the dynamics and the global reward.

Final State Controller (FSC): Although, any POMDP policy may be represented by a policy schedule, for some policies of an infinite horizon, infinite policy schedules may be required[4]. Therefore, most policy-based algorithms limit their search to finite political graphs, i.e., FSC which can be defined as an extension of a probabilistic automaton $\langle N, A, y, b_0, E\rangle$[5] along with the probability distribution $x: N \times A \rightarrow [0, 1]$ and the output set O.

Where:

N	=	Finite set of internal states of the controller
A	=	Course of actions POMDP
O	=	Set of observations POMDP
$y: N \times O \times \rightarrow [0, 1]$	=	State-transition function
y(n, o', n') = Pr(n'|n, o')	=	Probability of transition to n from the state n after observation o'
b_0	=	default beliefs
$E \subseteq N$	=	Discrete set
$x: N \times A \rightarrow [0, 1]$	=	Action choice function
X(n, a) = Pr(a|n)	=	Probability of taking action $a \in A$ in the state $n \in N$

Since, the concept of FSC is not accepted, E usually is not indicated. Moreover, ε-optimal FSC for average POMDP does not depend on the initial opinion b_0. Therefore, the definition of FSC can be reduced to $\langle N, A, O, x, y\rangle$ where A and O are known with this POMDP. This leaves only two unknown variables: x and y.

At each step t FSC takes $a \in A$ as entering the state and generates an observation o' in the next step in response. The transition of the internal state is probabilistic and is determined by recent history, as can be seen from the definition y. The number of internal states |N| represents the size of the FSC as well as the amount of agent memory[4]. Internal states are fully checked by the agent during the decision-making process, which selects the action to be taken on each node. $n \in N$ in relation to function x.

Sequential Qquadratic Programming (SQP): SQP is one of the most successful methods for solving problems of nonlinear limited optimization. It consists of a set of algorithms, not just one algorithm and is based on a deep theoretical foundation. SQP has demonstrated excellent performance in solving general problems of large-scale non-linear programming. In this study, we look at the following NLP problem:

$$\begin{aligned} &\min f(x) \\ &\text{s.t.}\quad g(x) \leq 0 \\ &\qquad h(x) = 0 \\ &\qquad x \in R^n \end{aligned} \tag{5}$$

where, x is a vector from n component; $F:R\to R^n$ objective functional; The functions $h(x):R^n\to R^m$ and $g(x):R^n\to R^l$ are resp. equality and inequality constraints.

SQP solves NLP to convert it into a series of problems with Quadratic Programming (QP). At each iteration, the original NLP is reformulated as a subtask QP, linearizing the constraints and replacing the objective function f(x) with its local quadratic approximation. QP subtask is:

$$\begin{aligned} \min\ & \frac{1}{2}d^T B_k d+\nabla f(x_k)^T \\ \text{s.t.}\ & \nabla g(x_k)^T d+g(x_k)\le 0 \\ & \nabla h(x_k)^T d+h(x_k)=0 \\ & d:=x-x_k\in R^n \end{aligned} \tag{6}$$

$$\nabla f(x_k):=\left(\frac{\delta f(x_k)}{\delta x_k^1},\frac{\delta f(x_k)}{\delta x_k^2},\dots,\frac{\delta f(x_k)}{\delta x_k^n}\right)$$

stands for the gradient function $f(x_k)$ at the point $xk = [x^1_k, x^2_k, \dots, x^n_k]$ and $B_k := Hf(x_k)$ is Hessian matrix f(x) at the point x_k that is the matrix of second partial derivatives of a function f(x). Assuming that x_k is a solution to the QP routine for k iteration which is actually an evaluation of the original NLP solution. So, the sequence $\{x^k\}_{k\in N0}$ converges to local optimal x* NLP. The basic idea of SQP is similar to the methods of Newton and quasi-Newton. However, the presence of constraints makes the analysis and implementation of SQP methods much more difficult. There are many NLP for which individual SQP methods exist to solve them. These NLP include unconditional optimization systems, linearly limited optimizations and non-linearly limited optimizations. We speak POMDP as NLP and rely on SQP tools to find solutions.

RESULTS AND DISCUSSION

Anonymity protocol analysis: To use the z-test, let us offer a simple algorithm for operational testing of the consistency of observations. The algorism defines whether the built pattern will statistical represent the data flow in the collection process. First, we collect a sequence of observational data y of some length D and build a model from the collected data. With a built model, we define z-statistics and find if experimental statistics provides 100 (1-α%) confidence that the transition with probability ε does not occur. If y is not long enough, we will not be able to build a model from the data; it is necessary to collect additional data. The algorithm is presented below.

Algorithm: Let us denote the transitional probability δ, when the system is in state s:

$$\gamma_s^\delta=\frac{\varepsilon}{\pi_s} \tag{7}$$

where, π_s asymptotic probability of the state s is defined by the formula $\pi_s\approx n_s/D$ wherever, n_s the quantity of times the condition s is introduced upon the supervision time y^D. Standardized z-statistics for the condition s is defined by the formula:

$$z_s=\frac{\gamma_s^\delta}{\sqrt{\frac{\gamma_s^\delta(1-\gamma_s^\delta)}{n_s}}} \tag{8}$$

Test z-statistics for the state s is defined by the formula:

$$z_{exp}=\min_s z_s \tag{9}$$

The model certainty is defined as:

$$\alpha_f=1-\prod_{s\in S}(1-P(Z<z_s)) \tag{10}$$

where, $P(Z<z_s)$ the probability that a normal distribution matters less z_s. Minimum amount of training data:

$$D^\alpha=\frac{n_s}{\pi_s} \tag{11}$$

Algorithm:

```
Input:
- repeated observation $y_t$ of the time t; -alphabet O; -threshold set by the user ε and the significance level α.
Output: -significance of the model αf
- demand  yardage |y|D
    Of  period factor t = 0
    (1) To build the $G_t$ model out of sequence y = $y_0$ $y_l$, ... $y_t$
    (2) To calculate the probabilities of the asymptotic state $\pi_s$ for $\forall s\in S$
    (3) Under (7) determine the values $\gamma^\delta_s$ for $\forall s\in S$
    (4) To calculate experimental statistics $z_s$ for each state (8)
    (5) To find $z_{exp}$ according to Eq. 9
    (6) If, $z_{exp}>z_a$ to conclude that $G_t$ is the provides act with the wishful layer of reliability and D = |y|
    (7) To compute αf according to equalization'(10); belay
    (8) Otherwise, gather larger information | |y| = $D^\alpha$ wherever |$D^\alpha$ is computed by equalization (11)
    (9) Go to step (1)
```

Summing up, we create the next guess-work about the supervision information and our lore of the underlying act. First, the act in question has a ultimate numeric of conditions and the conversion possibilities are static. This suppose guarantees that the learning information kit completely reflects the act. In addition, the alphabet O is completed and comprise every anticipated supervisions. Suggesting this, our treatment is limited to search "known unknowns"[6] at a given degree of statistic likelihood α. If the supervision is not in the alphabet, i.e. is an "unknown unknown"[6], the conversion not imposing the

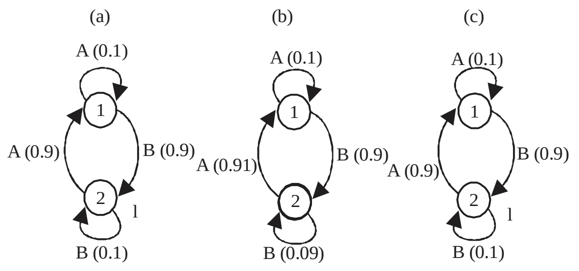

Fig. 3(a-c): Sample: (a) Original model; (b) Model built of 10 000 packages and (c) Model built of 100 000 packages

trust or the possibility of an anon conversion. Also, if $K_s = O$ and $U_s = \varnothing$ then the state has no possible fail-safe outgoing transitions. Conversions are not available to outlet the condition and state testing does not change the confidence in the model.

POC: The experiments presented in this section are simple. Our goal composed in that, to check the HMM formation algorithm[7] and allow some plain exemplum of the conception of pattern assurance. Below we consider the details of the application of the suggested algorithm for determining the model confidence in the detection of the Tor network protocol.

Sample: The HMM utilized to generation the supervision chain of sample is shown in Fig. 3a. Let's start with a random selection of the initial condition in this pattern. At every step, happens conversion occurs with an appropriate possibility and the appropriate character is observed. The initial act is configured to generate 10.000 data characters, Fig. 3b. We can see that the remodelled pattern has the same condition framework and nearly the same conversion possibilities as the initial pattern. When the act is repeated for 100.000 characters, we find that the possibilities correspond to the initial pattern.

Protocol detection: Now, we use the model trust approach presented above to determine the protocol that the sender uses when talking to a client over the Tor network, collecting time intervals between packets on the client. The time between sending each packet depends on the symbol associated with the transition. Each character is assigned a specified time delay in milliseconds and the server waits for this amount of time before sending the packet to the client. This method links inter-packet delays with HMM transitions. In other words, the time delays among serial packages will be our supervisions of the main act. This is the action that we anticipate in actual protocols that the package time will be associated with the handling demanded by a particular problem in this act.

Tor is a low-level overlay network that allows applications to communicate anonymously and securely on the Internet. An overlay network is a logical network connected by virtual circuits on top of a physical network. Links that connect individual systems in the overlay network are implemented as "tunnels" through the core network. Sent packets are encrypted multiple times so that they remain logically separate from normal traffic. The stability and explication of Tor can be explained by its pragmatic styling[8].

Tor basically consists of computers serving two types of services: a repeater and a directory server. There are several thousand relays, also known as onion routers, which operate on a voluntary basis by individuals and organizations around the world. The path through Tor is built of a relay. Relays and clients exchange data according to the catalog for the exchange of catalog information. By default, relays listen on TCP port 9001 for incoming requests. Active relays publish their router handles to the list of predefined directory servers (organs), reporting their current status. Directory servers store router handles on the relay list and constantly check the availability of these relays. In addition, each flag is assigned a different flag in accordance with their knowledge of the network status, that is which ones should be displayed as working, valid, stable, etc. Directory servers exchange their views with each other on the network on a regular basis, for example, every hour. After all servers match the list of available relays which is called consensus over the network, the consensus is published on the TCP port (default 9030) and available for download.

To use Tor, the client will need an HTTP proxy to retrieve the Tor directory and an HTTPS proxy to receive the relay. The current version of Tor allows the client to use any HTTPS or SOCKS proxy server to access the Tor network. Once installed, Tor can be initiated as an Onion Proxy (OP) if it processes only local requests. SOCKS proxy listens on port 9050 by default for streams created by TCP-based applications such as web browsing, SSH, instant messaging, etc. Then, the traffic will be routed via. Tor.

Tor starts building charts as soon as they have enough directory information. When the application flow arrives, it will be connected to a pre-built circuit, if it exists or wait until the circuit is available. Before building the circuit, the client selects all relays (by default by default) to use the launch with the output node. The entry node of the circuit must be one of the entry guards which is a set of nodes used by the client as long-term entry points to Tor.

The connection between the client and the entry node is first established using TLS/SSLv3 for authentication and encryption. After creating the first connection, the path extends to the second and third nodes in a similar way. Using this incremental path-building project, the client sets the session keys with each subsequent node independently. The final node of the scheme, known as the output node is selected to ensure, at best, support for connections to the destination.

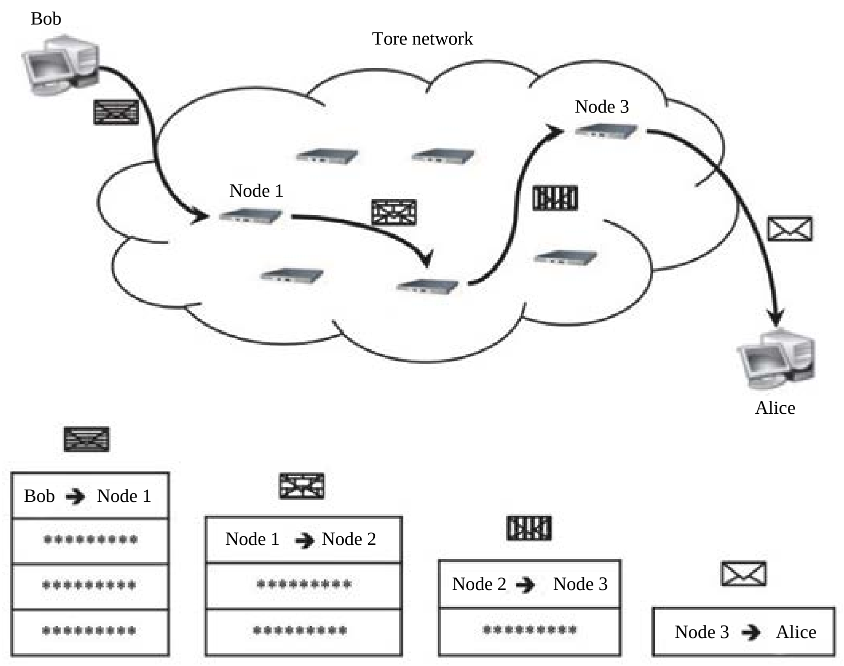

Fig. 4: The Tor cascade which originated from Bob, intended for alice is sent through a Tor circuit consisting of 3 relays

Before joining a stream to a built scheme that can support a client request, Tor will send a test request. If the request is not completed, Tor will send an error to the user.

All traffic going down the scheme is packed into 512-byte cells which is an effective measure against leakage of packet size information passing through the side channels. Then these cells are interactively encoded utilization the clef of every serial relay circuit. That is, the outermost layer of the packet is encrypted using the public key of the input node.

And so on, the innermost level of encryption is performed through the key of the output node. When a cell moves down the chain and comes to each relay node, the node "expands" the cell with its private key to identify where it should send the decrypted cell, for example, clear the onion skin. Thus, each node in the chain knows only the ascending node and the node downstream and cannot evaluate the entire panorama of the circuit. Thus, the compromise of a single node does not violate anonymity.

The procedure described is illustrated in Fig. 4 and 5. When the addressee, Alice, responds to Bob's request, the same process is performed in the reverse order. There are many other details of the process, such as encryption schemes, integrity checking, congestion handling, path selection, etc. A detailed specification of the Tor protocol can be found.

Here, is a pragmatic sample of detecting a protocol tunnelled via. Tor to illustrate the usefulness of the application of the suggested algorithm based trust model.

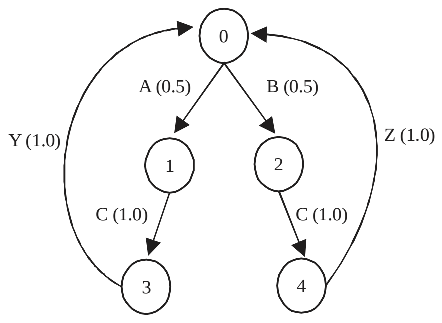

Fig. 5: Master five condition pattern for the pruning trial-trip

We utilize the treatment entered to derive a protocol pattern that the server utilizes at conversation to a customer via. the Tor network, by assembly time intervals between packets on the client.

First, we have a real HMM that introduces the protocol used. The time among despatching every package hinges on topical handling and is represented by the character connected with the conversion. Every character is appropriated a time setback band in milliseconds and the server waits for this quantity of time before sending the package to the customer. This method links inter-packet delays with HMM transitions. In other words, the time setbacks among serial packages will be our supervision of the main act. In actual protocols, the package time will be connected to the handling demanded by a particular problem in the act. Post

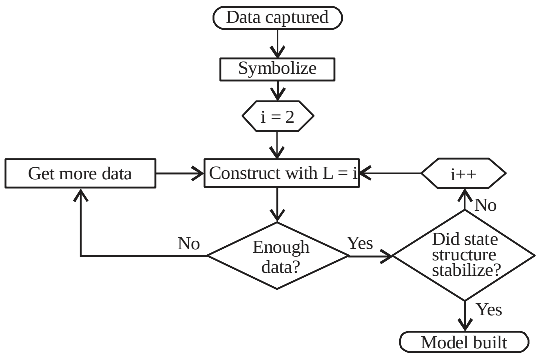

Fig. 6: Flowchart summarizing the process of building a model

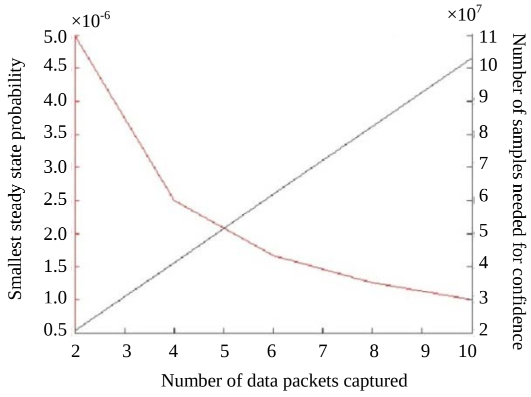

Fig. 7: Plot of model confidence results as more data is captured

designating the information that we record, the pattern building algorism is used to create the pattern utilized by the server.

Pattern building: The pattern utilized by the server in this trial-trip is shown in Fig. 5. The server launches the act by random selection a condition in its pattern as the beginning condition. To transmit every package, the conversion is taken from the actual condition and the suitable time setback waits before sending the package to the customer. If there is larger than one feasible conversion from a condition, the conversion is selected randomly, loaded by the possibility of every conversion. All information assembly was performed on acts transmitted via. Tor. The tshark program was utilized to taking package within the network. Calculate the distinction among every subsequent package time. We then represent the information, faction them into bands and assigning something in that band to a out-and-outer character such as or. We begin with and growth it as required. We abide by the act reported in the circuit in Fig. 6, to build the patterns demanded by our charge.

When the confidence test is escape on the pattern, we find that it requires 20,624,750 information examples. This means, we need to capture more data and rebuild. Because the quantity of demanded information is so, great, it has to be generated in run of 200,000 package at a time. post every kit of 200,000, we rebuild the pattern and run the assurance test over.

Strangely quiet, the demanded quantity of information keeps magnification with every kit. In a Tor compound are lost when a chain miscarries or shift or a relay becomes too occupied and setback a package. There is some additional rotational delay that manipulates abut one out of each 200 packages. These makes reason the package to come afterwards than it should have and because of that, it is wrong marked. All of these new occasion are so short possibilit which results in a below inferior limit asymptotical condition possibility for every new kit. This below possibility causes the assurance test to growth the volume of information demanded.

To prune these unsubstantiated states and transitions from the model, we use method of thresholding the asymptotic state probabilities.

Post a pattern has been built with CSSR, it may havetransitions that are is accepted very seldom and conditions that are visited very seldom. By installation a step on the asymptotical condition possibility, uncommon events are cased from the pattern. The clipped act is carried out basically in three steps:

- Each condition with an asymptotical possibility lower the sill is deleted from the pattern
- Any conversions running to or going out from that condition are too deleted
- Roundly, each condition or kit of conditions that cannot be achieved because of a disposal are too abstracted

This quit the pattern with alone the conditions and conversion for which we have fairly information. When we are we can to gather fairly information to be sure in the complete pattern, we disregard out the details where we would want more information to reach assurance.

The meaning of the possibility sill is how frequently, we should anticipate the act to digress from the pattern. The lower asymptotical condition possibility and suitable outcome from the assurance test are charted contrary the tally of package seized in Fig. 7.

The stable growth offer, we will not readily to collect fairly information to recover the pattern surely. As for our trial-trip, analysis of the asymptotical condition possibilities shows a great rupture among 71 of the conditions and the other eight. The 71 have possibilities below lower 0.06% while the other eight have possibilities above 8.2%. That is a gap of over two orders of value. This division creates a well layer of significance for shaving. Following the pruning act, the pattern in Fig. 8 outcomes with a threshold of 0.01 (or 1%).

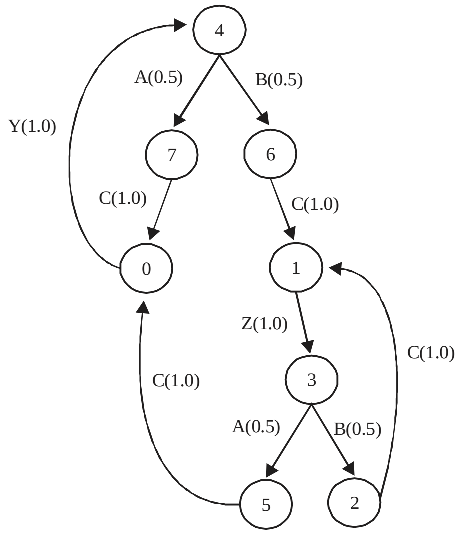

Fig. 8: Outcome post shaving deep-possibility conditions and conversion

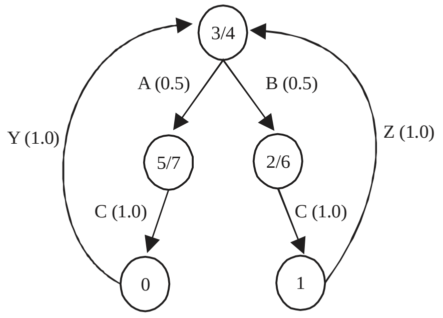

Fig. 9: The resulting model after the merging of states with the same probability distribution of the next output

It is appropriate at this point to comment that the conditions of the putative HMM are describe as having the same possibility allocation above the following output character. In this event, it abide by that nodes 3 and 4 in Fig. 8 being considered as one and the same condition and should be combined with each other. Similarly, nodes 5 and 7 are combined as well as states 2 and 6.

Nodes 6 and 7, albeit either have the identical output, must stay two individual conditions. Otherwise, the conversion guiding to the combined condition will be displayed on more than one character, that is on A and B. In Fig. 9 shows the abide by pattern which basically coincides with the eccentric pattern in Fig. 5.

CONCLUSION

Thus, the work describes the temporal side of the synchronization channel attack to detect a communication protocol tunnelled through Tor. algorithm based trust model is applied to the implementation of the attack. A proof-of-concept experiment on our private Tor network showed that a model could successfully be reconstructed from inter-packet timings and also proved the practical application of the algorithm based trust model.

REFERENCES

01. Rabiner, L., 1989. A tutorial on hidden Markov models and selected applications in speech recognition. Proc. IEEE, 77: 257-286.
02. Shalizi, C.R., K.L. Shalizi and J.P. Crutchfield, 2002. An algorithm for pattern discovery in time series. SFI Working Paper 02-10-060, Santa Fe Institute, Santa Fe, New Mexico.
03. Aras, R. and A. Dutech, 2010. An investigation into mathematical programming for finite horizon decentralized POMDPs. J. Artif. Intell. Res., 37: 329-396.
04. Braziunas, D., 2003. POMDP solution methods. University of Toronto-St. George Campus, Toronto, Canada.
05. Rabin, M.O., 1963. Probabilistic ?utomata. Inf. Control, 6: 230-245.
06. Rumsfeld, D., 2002. Department of defense news briefing. The News Transcript, Manalapan, New Jersey, USA.
07. Schwier, J.M., R.R. Brooks, C. Griffin and S. Bukkapatnam, 2009. Zero knowledge hidden markov model inference. Pattern Recognit. Lett., 30: 1273-1280.
08. Acquisti, A., R. Dingledine and P. Syverson, 2003. On the economics of anonymity. Proceedings of the International Conference on Financial Cryptography, January 27-30, 2003, Springer, Berlin, Germany, pp: 84-102.