Multi-Observer Privacy-Preserving Hidden Markov Models

Hung X. Nguyen and Matthew Roughan
School of Mathematical Sciences, The University of Adelaide, Australia.
E-mail:hung.nguyen, matthew.roughan@adelaide.edu.au

***Abstract*—Detection of malicious traffic and network health problems would be much easier if ISPs shared their data. Unfortunately, they are reluctant to share because doing so would either violate privacy legislation or expose business secrets. However, secure distributed computation allows calculations to be made using private data, without leaking this data. This paper presents such a method, allowing multiple parties to jointly infer a Hidden Markov Model (HMM) for traffic and/or user behaviour in order to detect anomalies. We extend prior work on HMMs in network security to include observations from multiple ISPs and develop secure protocols to infer the model parameters without revealing the private data. We implement a prototype of the protocols, and our experiments with the prototype show its has a reasonable computational and communications overhead, making it practical for adoption by ISPs.**

I. Introduction

As the Internet grows, and becomes more and more a part of the modern world's critical infrastructure, the issue of maintaining cyber-security confronts ISPs. There are many aspects to this problem, broadly falling under the categories of prevention, detection and mitigation. It may be ideal to prevent attacks before they can cause damage, but it is currently impossible to anticipate all possible attacks, and so detection of novel, or unexpected problems is necessary.

There is a now large literature on network anomaly detection (for examples see [4], [16]) aimed at detecting network problems. The range of techniques is large, but the general approach is to estimate the characteristics of the network under "normal" conditions, and then to look for substantial deviations from those characteristics. The principle differentiator in methods is the type of model used for normal conditions, and the method used to test for uncharacteristic behavior.

A number of authors have applied Hidden Markov Models (HMMs) to the task [1], [2], [13]. A Markov model is a simple stochastic process based on random, memoryless transitions through some series of states. In HMMs, we assume that the states themselves are not directly observable, but that we can make some indirect observations, and from these estimate the underlying process.

HMMs are among the most popular approaches for modelling time series data [10]. They have widespread applications in areas such as speech recognition, bio-informatics, and Internet traffic modelling [7], [14], and more pertinently to finding problems in networks [1], [2], [13]. Once we have trained such a model, we can then look for unlikely sequences of observations, and use these to signal anomalies.

There is no doubt that anomaly detection in general, and HMMs in particular, benefit from having as large a dataset as possible from as diverse a set of viewpoints as possible. In the Internet, these might take the form of observations from multiple ISPs who are all interested in detecting large-scale problems such Distributed Denial of Service (DDoS) attacks, worms, or address hijacking. However, such collaboration between ISPs is rare. The problem is that the type of data that must be shared is often considered sensitive; either because it contains business secrets, or customer or other data that has legal privacy requirements.

There are various approaches one could imagine to solve this problem. Here we apply an approach called variously Secure Multi-party Computation (SMC), or Privacy-Preserving Data Mining (PPDM). It has advantages over alternatives in that no-one (not even a "trusted" In effect, the entire computation is encrypted so that no participant learns anything except the desired answer.

In this work, we solve the problem of learning a HMM from observations made by multiple distributed and independent parties. The observations themselves are private; no-one can learn any one else's observations. However, there is almost no loss of fidelity as might be experienced in some anonymization schemes. The solution we obtain deviates only slightly from that if the data were completely public, provided some care is put into choice of key length and scaling coefficients.

Our solution performs computation in the encrypted domain to protect data privacy. Although data encryption and decryption introduce computation and communication overheads, we show in our experiments that even running on commodity hardware our prototype implementation can be used for realistic applications.

II. Background

A. HMM Theory

A Markov chain is a sequence of random variables $\mathcal{Q} = q_1 \dots q_T$ with the Markov property: given the present state, the future and past states are independent. Consider a Markov chain with N possible states $\mathcal{S} = \{s_1, \dots, s_N\}$. The Markov property is formally defined as

$$\mathbb{P}(q_{t+1} = s_i | q_1, q_2, \dots, q_T) = \mathbb{P}(q_{t+1} = s_i | q_t = s_j).$$

If the states of the Markov process are not directly observed, but rather we see some output sequence that is probabilistically associated with the Markov chain, the process is referred to as a Hidden Markov Model (HMM) [10]. A HMM is formally defined by the quintuple

- the set of N states $\mathcal{S} = \{s_1, \dots, s_N\}$,
- the set of M observation symbols $\mathcal{V} = \{v_1, \dots, v_M\}$,
- the initial probability
$$\boldsymbol{\pi} = (\pi_1, \dots, \pi_N), \text{ where } \pi_i = \mathbb{P}(q_1 = s_i),$$
- the time-independent state transition probability
$$\boldsymbol{A} = (a_{ij})_{N\times N}, \text{ where } a_{ij} = \mathbb{P}(q_{t+1} = s_j | q_t = s_i),$$
- the time-independent observation probability
$$\boldsymbol{B} = (b_{ik})_{N\times M}, \text{ where } b_{ik} = \mathbb{P}(O_t = v_k | q_t = s_i).$$

HMMs have been used successfully in detecting network problems [1], [2], [13]. In these applications, a HMM is built for normal traffic conditions. For example, HMMs are built in [1] for application protocols and in [2] for SSH traffic. Using these models, we can quickly evaluate the probability of an observed traffic stream. When an anomaly occurs, the likelihood of the observations will drop, and the deviation can be detected.

B. Introduction to Secure Multi-Party Computation

Secure Multi-party Computation (SMC) is a field of cryptography that provides means to perform arbitrary computations between multiple parties who are concerned with protecting their data. Mathematically, there are m parties $P_1, \dots, P_m$. Each party has private data x_i. They want to compute a joint function $(y_1, \dots, y_m) = f(x_1, \dots, x_m)$. The goal is to compute f without P_i learning anything about x_j or y_j for all $j \neq i$, other than what can be inferred form their own data x_i and output y_i.

The field of SMC originated from the work of Yao [15]. There is now a substantial literature on secure distributed computation and data mining. The parts most relevant to this work include work on applications to network management [3], [11], and anomaly detection using principle component analysis [5]; and application of SMC to HMMs [9], [12]. However our work is quite different from the last two cases, which consider the situation where one party holds the model, and the other the observations. That problem was relevant for a particular application, but has no obvious connection with network management. Instead, in our problem the observations are partitioned between the different parties.

We apply here the now standard *semi-honest* security model. In this model, the parties in the computation will follow the protocol correctly, but may perform additional (polynomial time) computations to attempt to learn additional information. Many of the techniques we use here have been extended to deal with adversaries who are willing to corrupt the protocol itself leading to invalid results, however, the resulting approaches are then more complex, and have larger overhead that is usually unnecessary for data-mining applications where it is in everyone's interest to find the correct solution.

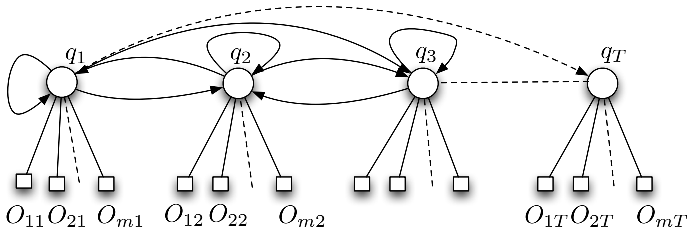

Fig. 1. A HMM with multiple observers.

The approach we take here is to use a number of well developed ideas from SMC as building blocks or *primitives* to create the algorithm we need. The challenge is to do so efficiently, and without intermediate data leaking useful information. The principle primitive used here is homomorphic encryption. We also briefly describe the other techniques used in this paper (secure logsum and secure negation).

Homomorphic Encryption: A homomorphic encryption scheme is a special type of public-private crypto-system with the property that some operation in the plain-text are mirrored by operations in the cipher-text. In practice, that means we can perform computations on the encrypted text, e.g.,

$$x \oplus y = D\big[E[x] \otimes E[y]\big],$$

for some operators $\otimes$ and $\oplus$. The homomorphic encryption scheme we use here is the Paillier crypto-system [8].

The other primitives we use are:

- *Secure logsum:* This is a simple protocol that uses the homomorphic properties of the Paillier crypto-system. Consider two parties A and B where A has a vector of encrypted values $(E[\log(x_1)], \dots, E[\log(x_m)])$, and B holds the decryption key. A and B want to jointly compute $E[\log \sum_{i=1}^m a_i x_i]$ for a public vector $(a_1, \dots, a_m)$. Algorithms for this protocol appear in [9], [12].
- *Secure negation:* We need to be able to allow two parties A and B to compute the encrypted negation $E[-a]$ of an encrypted value $E[a]$.

III. HMMs with Multiple Observers

Assume that we have a HMM with parameters $\{\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{\pi}\}$ as described in section II-A. There are m parties denoted by $P_1, P_2, \dots, P_m$ that make observations of the same underlying Markov process. Without loss of generality, assume that the observations are made between time 0 and T, and that the time interval is divided into T slots $[1, \dots, T]$. Each party makes its own observations of the system and these observations are secret. At each time slot, we have m observations of the underlying HMM. The observation set now becomes

$$\mathcal{O} = \{\mathcal{O}_1, \dots, \mathcal{O}_T\},$$

where each element $\mathcal{O}_t$ is a vector $\mathcal{O}_t = \{O_{1t}, \dots, O_{mt}\}$ of observations from each party. The sequence of T observations that party P_j makes is denoted as $\mathcal{O}^{(j)} = \{O_{j1}, \dots, O_{jT}\}$. An example of this model is given in Figure 1.

We shall assume that the observations of the different parties are independent. That is a natural assumption, as dependence between observations would weaken the need for privacy. Moreover, in this paper we assume all parties have the same observation probability given by the matrix $\boldsymbol{B} = \{b_{ik}\}$, though we plan to extend our model to the heterogeneous case in future work. Under these conditions the probability of a set of observations at time t, conditional on the state of the Markov process is given by

$$\mathbb{P}(\mathcal{O}_t|q_t = s_i) = \prod_{j=1}^{m} \mathbb{P}(O_t^{(j)} = v_{k_j}|q_t = s_i) = \prod_{j=1}^{m} b_{ik_j}.$$

We are interested in solving the training and evaluation problems for HMMs as defined in Section II-A in a privacy preserving manner. That is, the parties jointly compute the likelihood of the observations using the *forward* protocol and the HMM parameters using the *Baum-Welch* protocol [10] in such a way that at the end of the computation all parties learn the correct parameter value but do not learn anything more about the data of the other parties other than those that can be directly inferred from the output and their inputs. Details of the privacy-preserving *forward* and *Baum-Welch* protocols are provided in [6].

IV. Implementation and Results

In this section we describe our implementation of the secure protocols described in the preceding section. The code for our implementation is available at *www.hxnguyen.net*. It is written in Python, using the Paillier encryption scheme by Ivanov[1], which we have extended by adding the secure logsum and secure negation protocols. The HMM code was implemented on top of the standard HMM implementation by Hamiltom[2].

However, the code is not a trivial extension of these packages. The crucial implementation issue is that the encryption scheme applies to a finite field of non-negative integers, whereas the HMM algorithms were designed to work with potentially negative floating-point numbers. There is significant room for problems if the translation between these two domains is not performed carefully. We discuss details of the scaling and conversion in [6].

To test the performance of the secure protocols, we construct a simplified HMM for the detection of SSH brute-force attacks in [2]. In this model, the HMM has two states where the attackers alternate between "attack" and "inactive", i.e. $\mathcal{S} = \{Attack, Inactive\}$, and there are 6 observation outputs representing possible the traffic counts $\mathcal{V} = \{v_1, v_2, v_3, v_4, v_5, v_6\}$.

The following parameters are used for initial, transition and observation probabilities

$$\boldsymbol{\pi} = (0.01, 0.99), \quad \boldsymbol{A} = \begin{array}{c|cc} & attack & inactive \\ \hline attack & 0.95 & 0.05 \\ inactive & 0.05 & 0.95 \end{array},$$

$$\boldsymbol{B} = \begin{array}{c|cccccc} & v_1 & v_2 & v_3 & v_4 & v_5 & v_6 \\ \hline attack & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.5 \\ inactive & 1/6 & 1/6 & 1/6 & 1/6 & 1/6 & 1/6 \end{array}.$$

Key length (bits)	L	$\mathbb{P}(\mathcal{O}\|\boldsymbol{w})$	$\bar{\boldsymbol{\pi}}$	$\bar{\boldsymbol{A}}$	$\bar{\boldsymbol{B}}$
64	10^3	6.15 %	7.91 %	8.39 %	15.70 %
128	10^3	4.98 %	5.42 %	7.32 %	12.57 %
256	10^3	3.57 %	3.92 %	6.27 %	9.09 %
512	10^3	1.83 %	3.03 %	5.17 %	6.92 %
64	10^6	0.16 %	1.53 %	3.08 %	7.17 %
128	10^6	0.16 %	1.28 %	2.97 %	5.85 %
256	10^6	0.10 %	0.65 %	2.17 %	4.43 %
512	10^6	0.10 %	0.6 %	1.09 %	2.03 %

TABLE I
Worst-case errors over 10 runs, given as percentages.

We simulate a set of 10 realizations for the HMM, for each set of parameter values, and run both the centralized and secure distributed version of the estimation algorithms to compare.

A. Accuracy of the secure protocols

The approximation of real numbers by integers in the Paillier crypto-system introduces errors. Here, we compare the accuracy of the secure protocols against results provided by the ideal result as produced by a centralized algorithm. We compare the ideal result with the secure result using the relative error $\epsilon = |\hat{p} - p|/p$, where $\hat{p}$ is the result of the secure protocol, and p the ideal estimate. We evaluate the errors by varying the values of the scaling parameter L and the key length used in encryption.

Space constraints prevent us presenting results for each parameter $\bar{\pi}_i, \bar{a}_{ij}$ and $\bar{b}_{ik}$ so we present, in Table I, the *worst case* errors over each of the estimated components. The results (given for $T = 1000$ samples) show that the further through the estimation process we go (from $\bar{\boldsymbol{\pi}}$ to $\bar{\boldsymbol{A}}$ to $\bar{\boldsymbol{B}}$), the larger the errors, but that for large keys, and reasonable scaling parameters, the errors introduced by integer approximation and consequent over- or underflow are insignificant (2% in the worst case over multiple simulations, and parameter estimates). Larger keys also provide better security, so best performance occurs in the most secure case!

B. Runtime analysis

Another important consideration is the runtime of the protocol. We evaluate the runtime of our protocols by varying the key length and the number of samples used to evaluate and train the HMM, with $L = 10^6$. The results, generated on a laptop with a duo core 2.8 Ghz processor with 4GB of RAM are shown in Figure 2. The results show that the algorithm is approximately linear in the number of samples T, and quadratic as a function of key length. This quadratic growth is due to the runtime of the Paillier encryption and decryption functions, which could be more efficiently implemented. There is a clear trade-off between security and runtime as the longer the key length the more secure the protocol but computation time is also longer.

The Baum-Welch protocol takes on average 5 times longer than the secure forward protocol for each iteration, so estimation/training component of running these algorithms repre-

[1] *https://github.com/mikeivanov/paillier*
[2] *http://www.cs.colostate.edu/hamiltom/code.html*

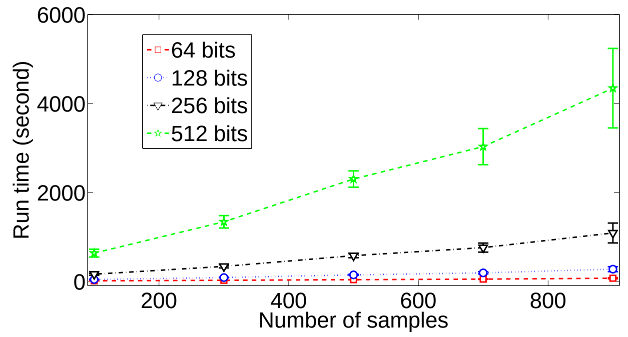

Fig. 2. Runtime for the secure forward protocol under different key-length.

sents a significant workload. However, detection of anomalies requires only the forward algorithm.

C. Collaboration benefits

The other obvious question to ask is whether there is an advantage in multiple parties collaborating. It is intuitive that a larger set of data is beneficial, but it may not be obvious that these benefits outweigh the costs involved in participating in such a protocol. Here we study these to allow potential collaborators to determine the cost/benefits.

Using the same model, we compare the errors as we increase the number of participants in the protocol. We apply the Baum-Welch algorithm to $T = 100$ samples and compare errors in the estimates. In particular, due to space restrictions, we focus on the estimates of $\boldsymbol{B}$, which we can see from Table I are the hardest to estimate accurately, and we calculate the Mean Squared Error (MSE) over the matrix.

The resultant MSE is shown in Figure 3. The plot shows that there is a substantial increase in accuracy when two parties collaborate, but that the marginal improvement lessens with increasing numbers of participants in the protocol.

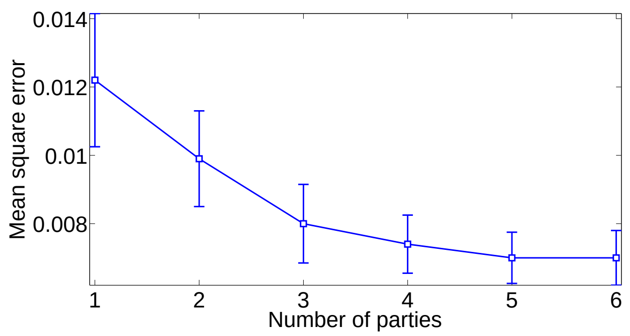

Fig. 3. MSE of the observation probabilities $\boldsymbol{B}$.

V. Conclusion and Future Work

In this paper we have shown that collaboration between multiple parties can improve the quality of estimates provided by HMMs. More importantly, we have shown how the parties can collaborate without revealing private data to each other. In the context of ISPs, this would mean that multiple ISPs can help each other detect network problems without running the risk of exposing critical data to competitors.

We have implemented the protocol using Paillier's homomorphic encryption. The implemented protocols are accurate and secure, and reasonably fast. However, as with all security there is a computational and communications overhead in encryption. In the future we plan to reduce this cost using more efficient algorithms.

There are other ways in which the protocol can be enhanced, for instance, we would use secure distributed protocols to prevent free-riding, which is hard to detect in the context of private data, and we aim to tackle this problem in future work.

References

[1] D. Ariu, G. Giacinto, and R. Perdisci. Sensing attacks in computers networks with Hidden Markov Models. In *Proceedings of the Machine Learning and Data Mining in Pattern Recognition - MLDM*, pages 449–463, 2007.

[2] C. Bartolini, L. Gaspary, A. Sperotto, R. Sadre, P.-T. de Boer, and A. Pras. Hidden Markov Model modeling of SSH brute-force attacks. In *Integrated Management of Systems, Services, Processes and People in IT*, pages 164–176. Springer Berlin / Heidelberg, 2009.

[3] M. Burkhart, M. Strasser, D. Many, and X. Dimitropoulos. SEPIA: Privacy-preserving aggregation of multi-domain network events and statistics. In *USENIX Security Symposium*, Washington, DC, USA, August, 2010.

[4] A. Lakhina, K. Papagiannaki, M. Crovella, C. Diot, E. D. Kolaczyk, and N. Taft. Structural analysis of network traffic flows. In *Proceedings of the joint international conference on Measurement and modeling of computer systems*, SIGMETRICS '04/Performance '04, pages 61–72, New York, NY, USA, 2004. ACM.

[5] S. Nagaraja, V. Jalaparti, M. Caesar, and N. Borisov. P3CA: Private anomaly detection across ISP networks. In S. Fischer-Hbner and N. Hopper, editors, *Privacy Enhancing Technologies*, volume 6794 of *Lecture Notes in Computer Science*, pages 38–56. Springer Berlin / Heidelberg, 2011.

[6] H. X. Nguyen and M. Roughan. Multi-observer privacy-preserving Hidden Markov Models. Technical report, University of Adelaide, 2011. http://www.hxnguyen.net/papers/TR_HMM.pdf.

[7] H. X. Nguyen and M. Roughan. SAIL: Statistically Accurate Internet Loss Measurement. In *Proceedings of ACM Sigmetrics 2010 Conference*, New York, NY, June, 2010.

[8] P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *ADVANCES IN CRYPTOLOGY EUROCRYPT 1999*, pages 223–238. Springer-Verlag, 1999.

[9] M. Pathak, S. Rane, W. Sun, and B. Raj. Privacy preserving probabilistic inference with Hidden Markov Models. In *Proc. of IEEE ICASSP 2011*, 2011.

[10] L. R. Rabiner. A tutorial on Hidden Markov Models and selected applications in speech recognition. *Proc. of the IEEE*, 77(2):257–286, February 1989.

[11] M. Roughan and Y. Zhang. Secure distributed data-mining and its application to large-scale network measurements. *ACM SIGCOMM Computer Communication Review*, 36(1):7–14, January 2006.

[12] P. Smaragdis and M. Shashanka. A framework for secure speech recognition. *Audio, Speech, and Language Processing, IEEE Transactions on*, 15(4):1404 –1413, may 2007.

[13] Y. Song, S. Stolfo, and T. Jebara. Markov models for network-behavior modeling and anonymization. In *Technical reports-Columbia University, http://hdl.handle.net/10022/AC:P:10682*, 2011.

[14] C. V. Wright, F. Monrose, and G. M. Masson. On inferring application protocol behaviors in encrypted network traffic. *J. Mach. Learn. Res.*, 7:2745–2769, December 2006.

[15] A. C. Yao. Protocols for secure computations. In *Proceedings of the 23rd Annual Symposium on Foundations of Computer Science*, SFCS '82, pages 160–164, Washington, DC, USA, 1982. IEEE Computer Society.

[16] Y. Zhang, Z. Ge, A. Greenberg, and M. Roughan. Network anomography. In *Proceedings of the 5th ACM SIGCOMM conference on Internet Measurement*, IMC '05, pages 317–330, Berkeley, CA, USA, 2005. USENIX Association.