Markov Inference Attacks

Jamie DeMaria
Advisors: Seny Kamara, Tarik Moataz

1 Abstract

We present and characterize a new class of inference attacks, called Markov Inference Attacks. We describe one such attack that relies on the Baum Welch algorithm for reconstructing hidden Markov models and graph matching. We analyze the feasibility of the attack by analyzing the accuracy and runtime of the Baum Welch and graph matching steps.

2 Background

Encrypted databases allow for secure storage of data and encrypted search algorithms enable user to search their data without decryption. Together, they allow for increased security and privacy. However, encrypted search algorithms produce information visible to an attacker. There is a fundamental tradeoff between search algorithm efficiency and the amount of this leaked information. A clever attacker can associate this leaked information with information stored in the database with what is called an inference attack.

Cryptography does not provide a formal way to analyze the vulnerabilities posed by this information leakage. Instead we develop our own inference attacks to understand these vulnerabilities and suggest solutions.

We present a new class of inference attacks called Markov Inference Attacks (MIAs). We observe that search queries can be thought of as a time sequence and that a user's current query may be related to their next query. Thus, we can model search queries as a Markov model. With our Markov assumption, we have a new set of algorithms and analysis methods with which to attack and encrypted database and search algorithm.

3 The Attack

In our attack we have access to the leaked information produced every time a user queries the database. A single search keyword may produce different leaked informa-

tion with each new search, so we must model the leaked information as a hidden Markov model (HMM). We also have access to a known dataset of search query patterns. Our goal is to find a mapping between a piece of leaked information and a plaintext keyword so that we can know what the user is searching for.

Our attack works as follows: first we use the Baum Welch algorithm to reconstruct an HMM that likely produced the sequence of leaked information. We do the same with our known dataset of search query patterns. We focus on the state transitions of these two Markov models and shift our perspective to view them as weighted directed graphs rather than probability models. We now want to find the best matching of the leaked information graph in the known data graph. We do this using a graph monomorphism algorithm.

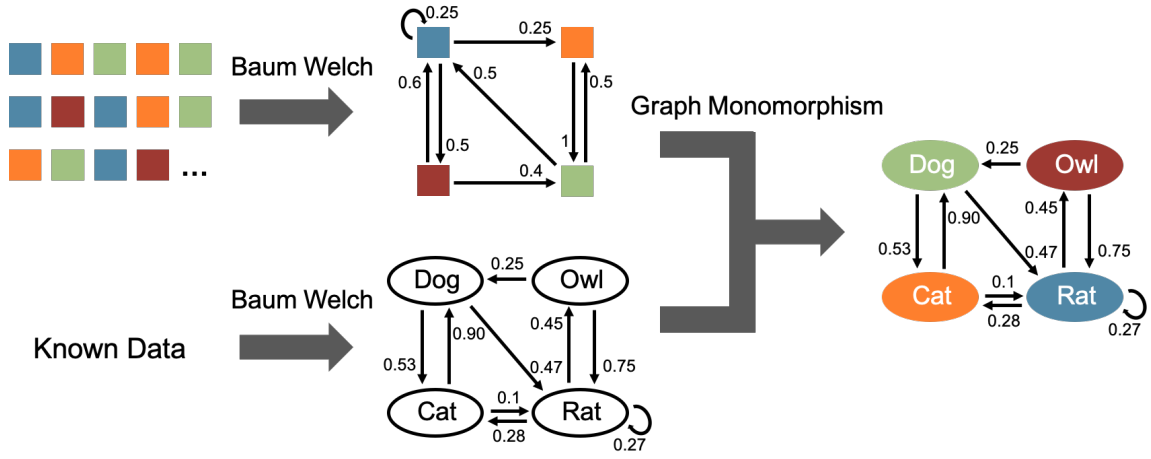

Figure 1: Graphic describing the attack sequence

For initial analysis of the feasibility and accuracy of this attack, we make two assumptions. First, that a single keyword produces exactly one piece of leaked information, and second, that the leaked data graph and the known data graph have the same number of states.

4 Baum Welch

The Baum Welch algorithm is a canonical Machine Learning algorithm for finding a hidden Markov model that has high probability of producing a sequence of data. We wish to analyze the runtime and accuracy of the Baum Welch algorithm in the context of our attack. We look at the effects of the number of unique states in the observed sequence, the length of the observed sequence, and the number of iterations the algorithm was allowed to improve its model before termination.

To measure the accuracy of a Baum Welch reconstructed HMM, we compute a distance metric between the reconstructed HMM and the original HMM that produced the observed sequence. We do this by computing the average error between corresponding edges of the two models. We call this metric the distance between two HMMs. Distance is the inverse of accuracy.

4.1 Results

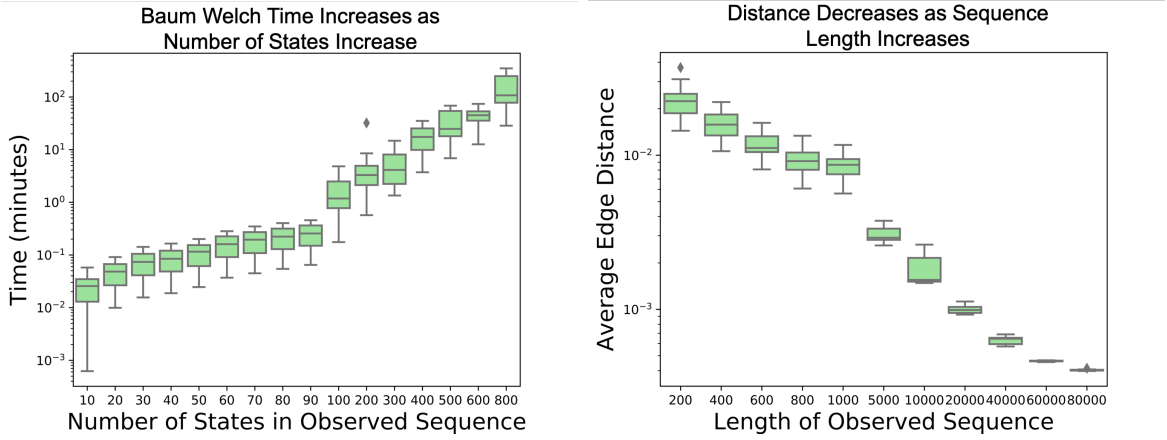

Figure 2: Baum Welch experiment results

We found that the number of states in the observed sequence has the most pronounced effect on runtime. A larger number of states results in a longer runtime. The length of the observed sequence has the most pronounced effect on accuracy. A longer sequence results in a smaller distance. The Baum Welch algorithm requires high amounts of memory. When running the algorithm on observed sequences with 10,000 states, over 64 gigabytes of memory were required.

The number of iterations the algorithm could improve the HMM did not have an effect on runtime or accuracy.

5 Graph Matching

We want to find a matching between a Baum Welch reconstructed HMM and a Markov model produced from our known data. Finding an optimal graph isomorphism seems like it will produce this matching. However, graph isomorphism requires a bijective matching between the vertices of both graphs and the edges of both graphs. Since we are not guaranteed to see every possible state transition in our observed sequence of leaked information, we cannot guarantee that all edges in the known data graph will be present in the leaked information graph. Thus we turn to a lesser known matching

scheme called graph monomorphism. Graph monomorphism only requires a bijective mapping between the vertices and an injective mapping of the edges in the leaked data graph to the edges in the known data graph. In the more general version of the attack when the known data graph may be much larger than the leaked information graph, we look for an optimal subgraph monomorphism.

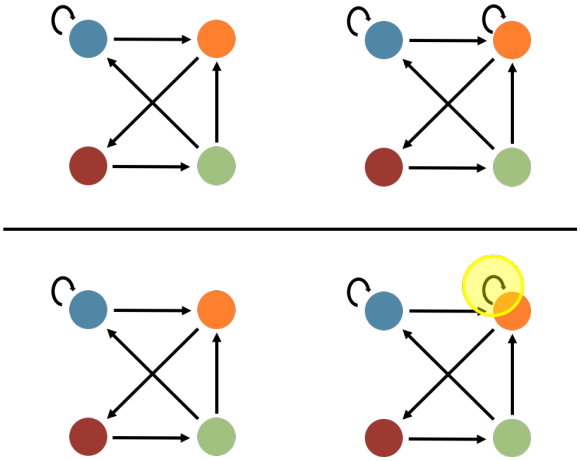

Figure 3: *Top:* We can see that the graph on the left is contained in the graph on the right *Bottom:* Because of the highlighted edge, there is no isomorphism between the graphs. Monomorphism allows us to ignore the highlighted edge and finds the matching between the two graphs

An algorithm for an optimal subgraph monomorphism algorithm is presented in [1]. We implement this algorithm and explore the effects of the number of states (vertices) in the graph, distance between the two graphs, and graph structure (ex. the existence of cliques in the graph) on the runtime and accuracy of the algorithm.

5.1 Results

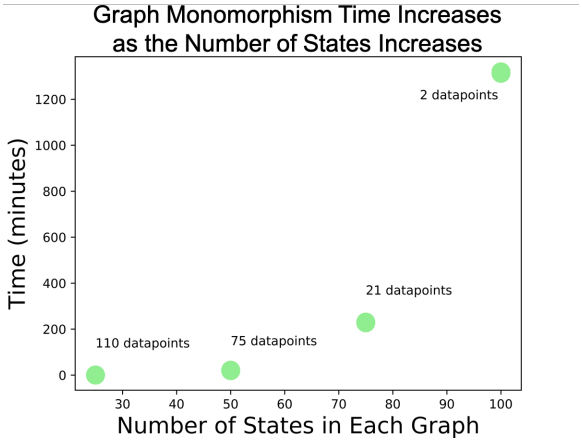

Figure 4: Graph Monomorphism experiment results

We found that the number of states in the graphs has the most pronounced effect on runtime. A larger number of states results in a longer runtime. Additionally, graph monomorphism with over 500 states takes over a week to complete.

The distance between the two graphs and graph structure did not have an effect on runtime or accuracy.

6 Conclusion

We found that that the runtimes to complete the Baum Welch and graph monomorphism steps of the attack are most affected by the number of states in our sequence of leaked information, with the graph monomorphism step taking the longest. We also found that the Baum Welch step requires a large amount of memory.

To make encrypted databases and search algorithms more secure to Markov Inference Attacks, we want to design them with the properties that make the attack difficult. These results tell us that artificially inflating the number of states in the leaked information is a promising way to make the attack infeasible.

Understanding and exploring potential improvement to the graph monomorphism step of the attack is an important component of our future work. Additionally, exploring the feasibility of the attack when our initial analysis assumptions do not hold and in more real world setups will allow us to further understand the feasibility and generalizability of this attack.

References

[1] S.C. Chan A.K.C. Wong, M. You. An algorithm for graph optimal monomorphism. *IEEE Transactions on Systems, Man, and Cybernetics*, 20(3):628–238, 1990.