IJCA, Vol. 24, No. 1, March 2017

Rewind: An Automatic Music Transcription Web Application

Chase D. Carthen*, Vinh Le*, Richard Kelley*, Tomasz J. Kozubowski*, Frederick C. Harris Jr.*
University of Nevada
Reno, Nevada, 89557, USA

Abstract

Simple digital audio formats such as mp3s and various others lack the symbolic information that musicians and other organizations need to retrieve the important details of a given piece. However, there have been recent advances for converting from a digital audio format to a symbolic format a problem called Music Transcription. Rewind is an Automatic Music Transcription (AMT) system that boasts a new deep learning method for generating transcriptions at the frame level and web application. The web app was built as a front end interface to visualize and hear generated transcriptions. Rewind's new deep learning method utilizes an encoder-decoder network where the decoder consists of a gated recurrent unit (GRU) or two GRUs in parallel and a linear layer. The encoder layer is a single layer autoencoder that captures the temporal dependencies of a song and consists of a GRU followed by a linear layer. It was found that Rewind's deep learning method is comparable to other existing deep learning methods using existing AMT datasets and a custom dataset. In other words, Rewind is a web application that utilizes a deep learning method that allows users to transcribe, listen to, and see their music.

Key Words: Deep learning, Automatic Music Transcription, Music Information Retrieval, and Machine Learning.

1 Introduction

Many musicians, bands, and other artists make use of MIDI, a symbolic music instruction set, in popular software to compose music for live performances, portability across other formats, and recording. However, most music is often recorded into raw formats such as Wav, MP3, OGG, and other digital audio formats. These formats do not often contain symbolic information, but may contain some form of metadata that does not typically include symbolic information. Symbolic formats, such as sheet music have been used by bands, choirs, and artists to recreate or perform songs. These symbolic formats are effectively the spoken language of music that can be re-translated back into sound. Communities such as Mirex are actively working many different problems on retrieving information from music so that creating, categorizing, and extracting information is easier. The Symbolic format is not only portable, but can be leveraged for doing different types of analysis such as genre classification, artist classification, mood detection.

Automatic Music Transcription (AMT) is the process of converting an acoustic musical signal into a symbolic format [14]. There are a few music transcription applications having varying degrees of accuracy that have been built mostly for Windows, Linux, Mac and the web browser [19]. Only a few of these applications have the ability to visualize the results of the transcription. A piano roll is an intuitive visualization of music that does not require a user to learn a more complex symbolic available for music such as sheet music. These applications allow a user to get a symbolic format of their music that can be used for many different reasons such as changing a song, portability to other applications, live performances, and for generating sheet music. However, most of these applications do not use state of the art algorithms from advances in Deep Learning that have contributed to the Music Information Retrieval (MIR) field.

There has been recent work in the AMT field with [5, 6, 25] that have produced higher transcription accuracies than previous methods. These advances along with the creation of web audio frameworks such as WebAudio or WebMidi have made it possible to playback many different types of audio formats such as mp3, wav, and MIDI. Web frameworks such as Django and Flask make it possible to create a web application that does automatic music transcription and allows users to visualize the transcription and hear the results. Rewind [8, 9] is a tool and method that will make use of a new Deep Learning method based on previous work, visualize the results of the transcribed file, and allow the user to edit the transcribed results.

The following paper is structured as follows: Section 2 covers background related to the MIR and Deep Learning field.

*chase@nevada.unr.edu, vle@nevada.unr.edu
*rkelley@unr.edu, tkozubow@unr.edu
*Fred.Harris@cse.unr.edu

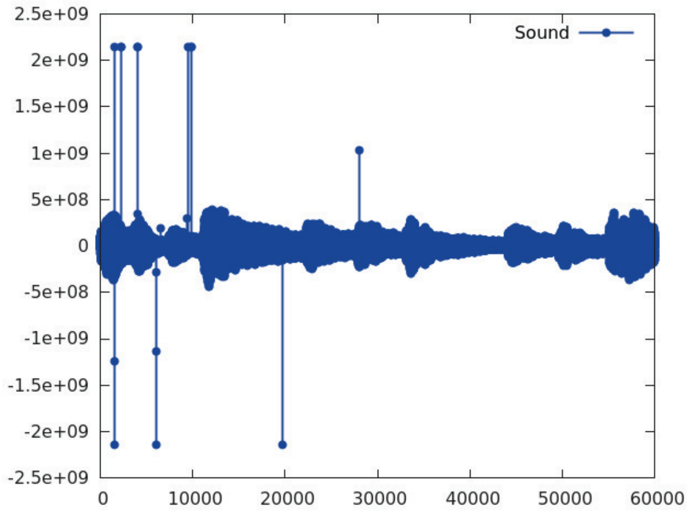


Figure 1: An example of a raw audio file

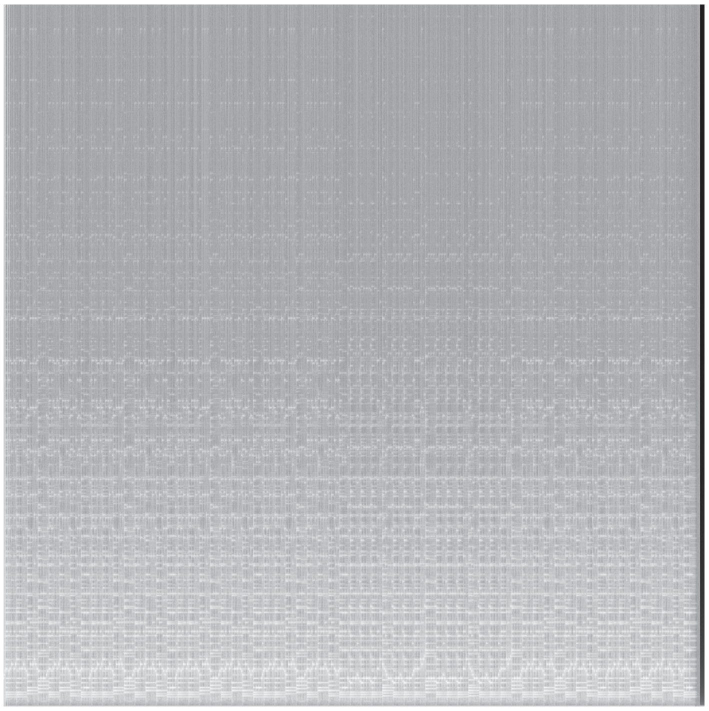


Figure 2: An example of a spectrogram

Section 3 discusses the implementation and design of Rewind tool. Section 4 gives the results of the Rewind method. Finally Section 5 concludes and details future direction that Rewind can take.

2 Background

AMT systems are designed to make transcriptions at the three levels of detail in music, those being the stream, note, and frame level [14]. The stream level is simply a raw acoustic signal which is contained in an audio digital file, an example of which can be seen in Figure 1. The goal of the frame level is to capture all pitches within each frame provided by a spectrogram. An example of a spectrogram is demonstrated in Figure 2. At the note level, a set of pre-existing notes are used to generate a brand new set of notes or create a record of the notes. The note level can be represented as a piano roll or as sheet music. An example of sheet music and piano roll is demonstrated in Figures 3 and 4. Most AMT systems evaluate their effectiveness by means of various metrics, including recall, accuracy, precision, and f-measure [3]. Precision determines how relevant a transcription is given irrelevant entries in a frame. It is defined as follows:

$$Precision = \frac{\sum_{t=1}^{T} TP(t)}{\sum_{t=1}^{T} TP(t) + FP(t)} \tag{1}$$

Recall is the percentage of relevant music transcribed, and is given by Equation 2.

$$Recall = \frac{\sum_{t=1}^{T} TP(t)}{\sum_{t=1}^{T} TP(t) + FN(t)} \tag{2}$$

The accuracy determines the correctness of a transcription, and is given by Equation 3.

$$Accuracy = \frac{\sum_{t=1}^{T} TP(t)}{\sum_{t=1}^{T} TP(t) + FP(t) + FN(t)} \tag{3}$$

While the F-measure determines the overall quality between the precision and recall.

$$F\text{-}measure = \frac{2 * precision * recall}{precision + recall} \tag{4}$$

These metrics in turn are calculated with true positives, false positives, and false negatives

There has been some work using LSTMs and semitone filter banks to transcribe music [5]. In Sigtia's work [25], the idea of an acoustic model converting an audio signal to a transcription is introduced. Additionally this paper introduces using a music language model to improve the accuracy of a transcription of a acoustic model like Boeck [5] and others as well. Boulanger-Lewandowski [6] uses a deep belief network to extract features

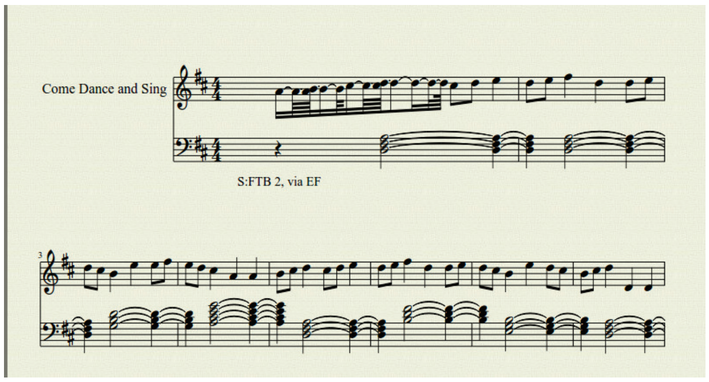


Figure 3: An example of sheet music

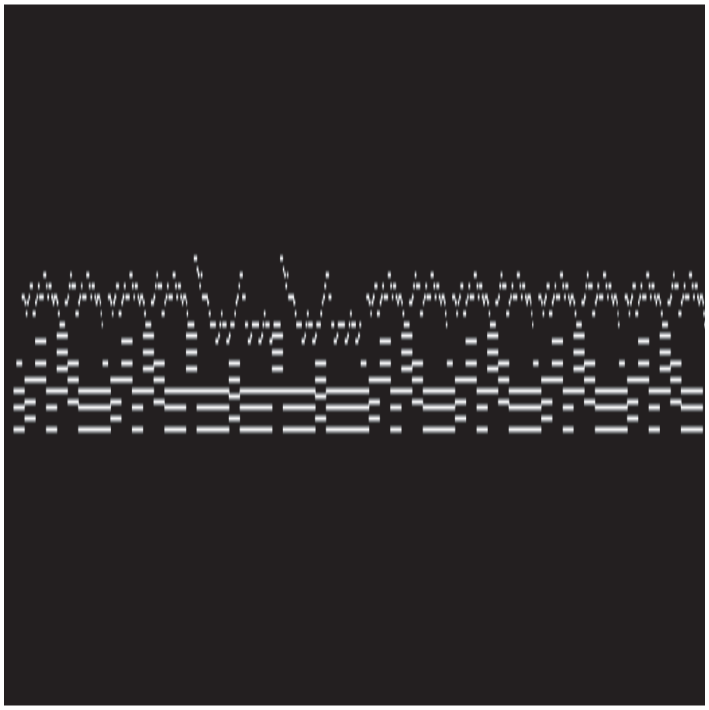


Figure 4: An example of a piano roll

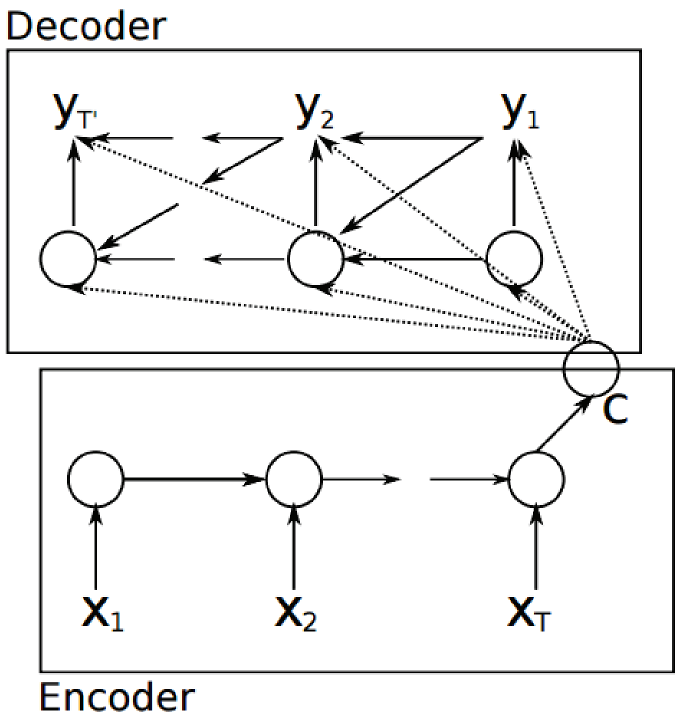


Figure 5: A picture of a encoder-decoder network with a context C demonstrated between the encoder-decoder network [11]

from a spectrogram and utilizes a rnn to create a transcription along with a innovative beam search to transcribe music. Boulanger-Lewandowski's beam search is possible thanks to the generative properties of the deep belief network that is merely a collection of restricted Boltzman machines or RBMs that are stacked. This beam search is also utilized in combination with recurrent neural network with an neural autoregressive distribution estimator (rnn-nade) as a music language model and an acoustic model that uses a deep neural network. A follow-up paper produces a hash beam search that finds a more probable transcription in fewer epochs [24]. Both the beam search and hash beam search produce the most accurate transcriptions.

Recently, encoder-decoder networks have been used for unsupervised learning in terms of autoencoders [26], translation [12], caption generation for images, video clip description, speech recognition [11, 13] or video generation. Autoencoders, like an encoder-decoder network, are commonly used for unsupervised learning to learn features contained inside the data, by using the identity of the data. An autoencoder is powerful for learning features contained within a dataset. However, there are more complex encoder-decoder networks [12, 11, 13], where they learn a context and then map English to French. They are less concerned with learning the identity and more for learning the context of the data presented. Rewind utilizes these types of encoder-decoder networks to learn an encoding for a spectrogram presented to it. An example layout of this network is demonstrated in Figure 5. These networks have proven to be beneficial, and are state of the art.

3 Rewind

Rewind is very much like other AMT systems in that it determines the fundamental frequencies of the notes and what notes are active at the frame level. Like most other frame based systems, Rewind utilizes a spectrogram as its main input and a ground truth midi as the target. All audio samples are constructed at a 22 kHz sample rate and turned into a normalized spectrogram with a 116 ms window size, which can be either a 10 ms or a 50 ms stride. It has been found that a window size larger than 100ms produces the most accurate results with a rnn-lstm [5]. A multitude of existing datasets were utilized for training Rewind's models: Nottingham [1], JSB Chorales [2], Poliner and Ellis [20], Maps [15], MuseData [10], and Piano.midi.de [16]. All of these datasets were split into 70% for training, 20% for testing, and 10% for validation. These datasets consisted of midi only or midi with aligned audio and made into datasets with timidity, Torch's audio library, and a midi library [4]. Rewind's models were implemented with rnn [18] and optim. A simple auto-correlation method was also constructed as a way to implement Rewind's web service and website for quick testing. The auto-correlation is also compared against the encoder-decoder network. Rewind has two types of models: the encoder and the decoder model. The encoder and decoder is very similar to the encoder-decoder network in Figure 5 [11, 12, 13]. The encoder model of Rewind utilizes an autoencoder, which utilizes a single GRU for its encoder, whose

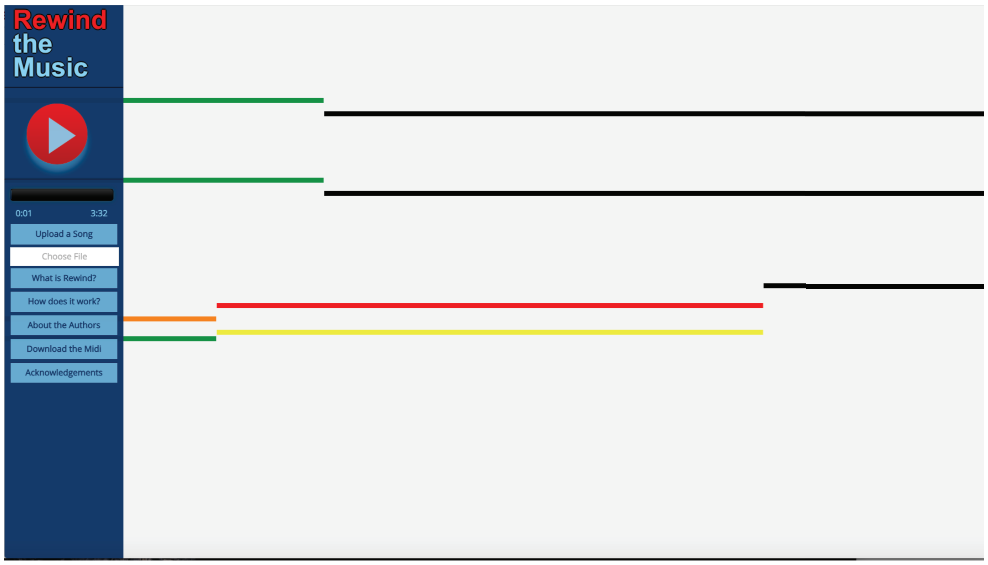


Figure 6: A screenshot of piano roll notes lighting up

output is squashed by a rectified linear unit and a linear layer for its decoding layer. While the decoder model has an identical layout, but its outputs are squashed with a sigmoid activation function and may have a second GRU in parallel.

The encoder network utilizes an autoencoder to create an encoding for a spectrogram. An autoencoder was chosen because a deep neural network (stacked auto encoders) has been used for extracting features from a spectrogram in the case of speech recognition [7] and other similar works that utilize deep belief networks (stacked restricted Boltzman machines) have been used to extract features [17]. A deep belief network, along with an autoencoder, are used to produce a generative model for a spectrogram [13]. The generic representations generated by autoencoders can be further improved with recurrences [26], where the encoder and decoder of the autoencoder are both LSTMs for learning over video sequences and generating video sequences. Rewind's encoder model utilizes a linear neural network for the decoder and a GRU for the encoder with a rectified linear unit (ReLU) for it's activation function [26]. The encoder network is trained with a mean squared error function.

The decoder network consists of two types of networks being a GRU with a linear layer and two GRUs stacked onto each other in parallel with a linear layer. Both types of networks are squashed with a sigmoid function. The GRU in both networks was chosen because it produced the lowest error rate. This network's objective function is binary cross entropy, so that this decoder network will learn a distribution of notes where a probability of one indicates a note on and a probability of zero indicates a note off. Binary cross entropy is used for minimizing the log probability [6, 25], which also utilizes a sigmoid function to create binary probabilities[23]. The binary cross entropy function is demonstrated in Equation 5, where the sum is taken over all distributions [25]:

$$\sum_i t_i \log p_i + (1-t_i)\log(1-p_i) \tag{5}$$

The probabilities constructed from the sigmoid function can be used to construct a MIDI, and are utilized in previously mentioned papers. The decoder network's job is to generate these probabilities for each encoding passed by the encoder network.

The auto-correlation method is a very noisy method. The process creates a spectrogram of the required audio file and then each bin of the spectrogram is normalized with the standard deviation and mean. After these transformations have been made, a threshold is applied, where anything greater than the threshold is a 1 and anything less is a 0. Subsequently, one simply only needs to go to each frequency bin that matches a midi note and extract the frequencies that have a value of 1. This auto correlation method is only meant as a test model for a web service. However, in Section 4, results are reported for its accuracy in comparison to Rewind's Network.

3.1 Architecture

Rewind's architecture consists of multiple parts that consist of: the client, models and web service, and the server. Each part is unique and has been designed to handle different parts of Rewind's functionality. The models are used for producing transcription, and the web service is used to interface with the model and send outputs to the client through the server. All visualization, downloads, and uploads are handled by the client. The server pushes all content needed to run the website to the client. An overall diagram of the architecture is demonstrated in Figure 7.

The models and web service component of the architecture are used to process data for training a model, generating

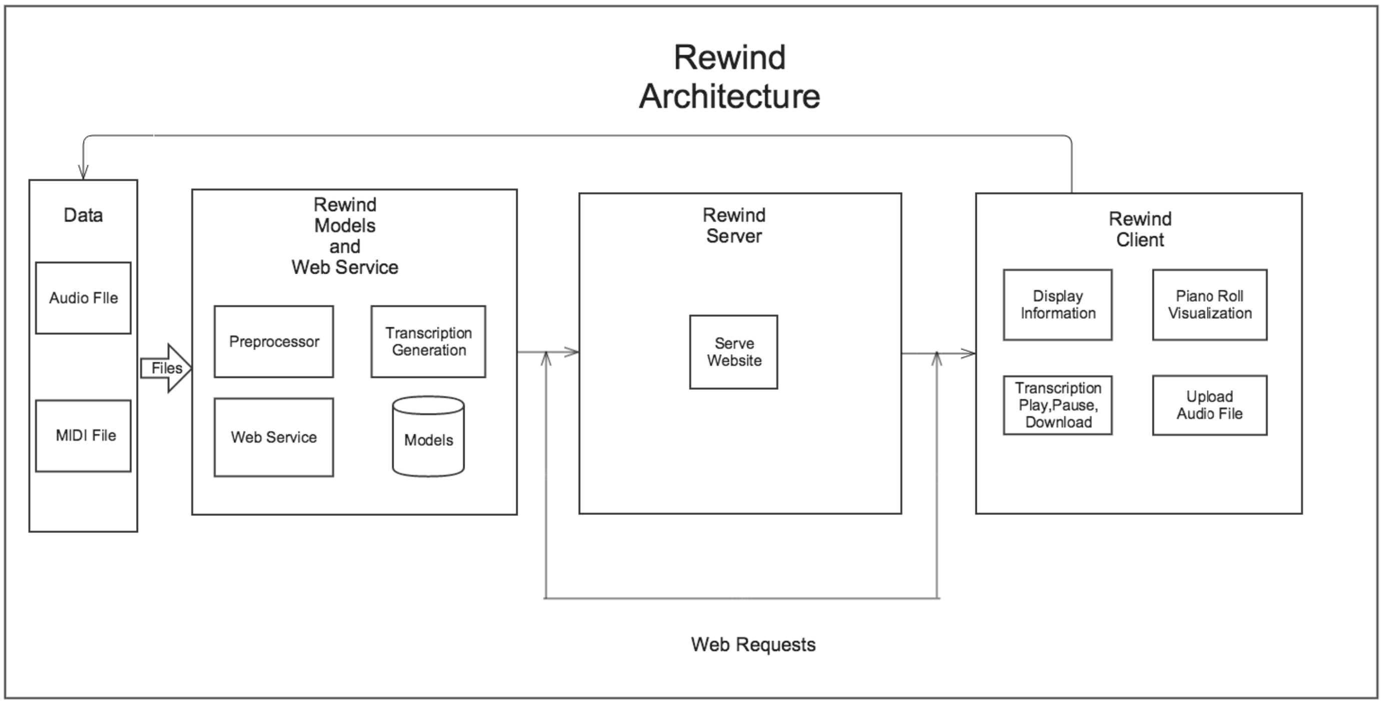


Figure 7: The Architecture of Rewind

transcriptions with a preexisting model to be sent through the web service, and training models. This component contains Rewind's method, or AMT algorithm, for creating transcriptions of digital audio formats. The web service was created as a way for Rewind's models to send transcriptions to the client. The web service for Rewind was written in Flask [22], as it requires a small amount of code to get a web service written.

Rewind's server was created with Django's web framework [21]. The rewind server serves up the website to the client, which includes all of the HTML, Javascript, and CSS files. It also handles sending uploaded audio files to the web service and forwarding the content back to the client.

The client is a web browser, such as Google Chrome or Mozilla Firefox, that is to be utilized by the user. The client handles creating a piano roll for visualization, uploading audio files to the web service, and giving the ability to download a transcription. All sound playback is handled by the client and allows the user to pause and play sounds. The client's job is to light up the notes in the piano roll as the note on hits.

3.2 Use Case Modeling

This section describes the use cases of Rewind and covers the different scenarios of Rewind. The use cases were created to understand what the user needs are for Rewind. Both the back end of Rewind, being the models and web service, and the front end of Rewind, being the Graphical User Interface (GUI) of Rewind or the client, are covered by these use cases.

In the full use case diagram shown in Figure 8, there are four actors being the: User, Developer, Web Service, and the Rewind Server. The User are those who are interested in creating a transcription of a digital audio song. The Developer is one whom that is expanding and/or improving the accuracy of Rewind. The Web Service is a service that allows the Rewind client to convert a digital audio format into transcription. The Rewind Server serves a website to the Rewind client. The rest of the section explains each use case of Figure 8.

Play/Pause Playback
The user has the option to pause or playback a given transcription in the Rewind client.

Download Transcription
When a transcription has been received from the server, the user may download a transcription that one had requested.

Inspect Piano Roll
The user may look around the piano roll within the Rewind client.

Get Information About Project
The Rewind client will provide the user the option to get information about the Rewind project and how the project works.

Upload Audio File
The user in this use case will upload a file that they wish to transcribe.

Receive Transcription
When the server has received a transcription from the web service, the Rewind client will receive the transcription for playback and visualization.

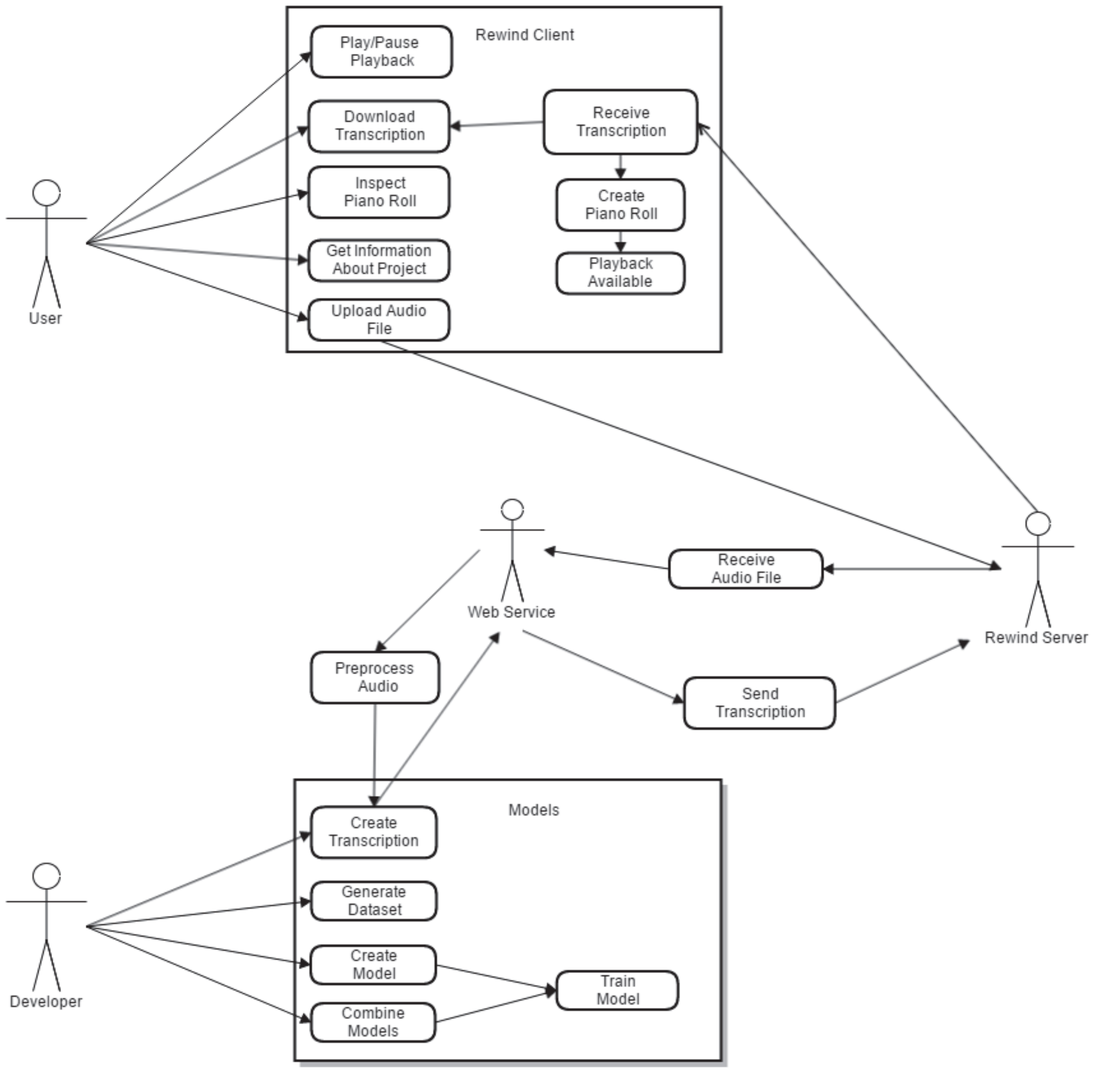


Figure 8: A Use Case Diagram of Rewind

Create Piano Roll
After receiving the transcription the rewind client will build a piano roll transcription for the user to see.

Playback Available
After the piano roll has been inside of the Rewind client, then the client will allow the user to playback the transcription and will let the user know that playback is available.

Receive Audio File
In this use case, the web service receives an audio from the server and is now ready to preprocess the audio file for transcription by the models.

Create Transcription
The create transcription use case can occur in two different ways: one is when the web service sends an audio file to the models for transcription or a developer invokes the service.

Send Transcription
When the models have finished transcribing, then the transcription will be sent to the web service where the Rewind server will then send the data to client.

Preprocess Audio
The models before they can transcribe any audio have to make sure that the files themselves are the proper format. If they are not proper, then by default the models will transform the music into the proper format.

Generate Dataset
The developer may wish to generate a new dataset for training the models, which is possible. This is so the developer may tweak Rewind and make its overall transcription accuracy better.

Create Model
The developer is also able to create new models that can be

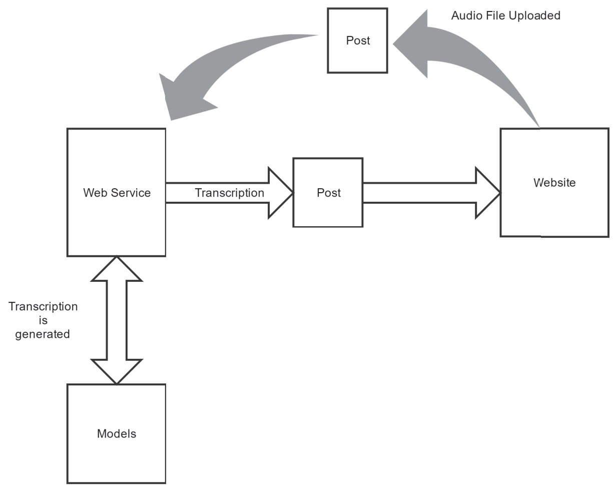


Figure 9: A diagram of Rewind's web service

utilized for transcription or research.

Combine Models
The developer may wish to combine multiple models together in order to improve transcription.

Train Models
The developer has the option of training the models in order to determine if the new model is better than the current model utilized by the web service.

3.3 Website and Web Service

Rewind's web service was implemented in Flask as a small web service that could be utilized by Rewind's server for making transcriptions of uploaded audio files. A small web service was implemented for transcribing audio files so that Rewind would remain scalable. All audio files and transcriptions are handled through post requests. Figure 9 demonstrates a diagram of the communication of audio files and transcriptions going in and out of the web service. This web service communicates with the models of Rewind and creates a midi file from the passed in audio file. All transcriptions generated by the web service are piano only. Rewind's website was implemented in the Django web framework and utilized the following javascript libraries: remodal, jQuery, jQuery UI, and midi.js. Django was chosen for Rewind because it allows Rewind to be scalable for future development, stable database integration, and future incorporation of security. Midi.js is utilized for its ability to parse MIDI files and generate sounds for those MIDI files. The jQuery and jQuery UI libraries has many useful features for designing interfaces, such as animations, element manipulation, and 3D effects. The remodal library allows for seamless modal windows to be displayed on the website. A small web service was implemented in Flask to wrap Rewind's models in order to be utilized by Rewind to generate transcriptions through http requests. This service was implemented so that the small web service could be independent and be used in other applications if needed. These libraries have made it possible to make a website for Rewind. An example of Rewind's website is demonstrated in Figure 6. This figure also demonstrates Rewind's ability to visualize the playback of a midi file in the form of a piano roll where the colors denote the note level. The user has the ability to scroll through the piano roll using the time bar and inspect the piano roll validity.

Rewind has a built in web synthesizer, which is used to playback transcriptions generated by Rewind's models. Midi.js has several dependencies, which are used to playback sounds and can handle different platform setups. It can parse midi events and make it possible to extract time delta for constructing piano rolls and note information. Midi.js can load many different sound fonts to load different sounds such as piano, flute, drums, and other sounds. The piano roll constructed for visualization in Rewind is based on the time duration and time position information collected from midi.js. The user has the ability to scroll through the piano roll using the time bar and inspect the piano roll validity. As a song plays the piano roll will light up each note with different colrs based on the note number as demonstrated in Figure 6, and the screen will transition to another part of the piano every second. There is some future work to be developed regarding the ability of adding or removing certain notes from the transcription using the piano roll. In conclusion, these libraries allow Rewind to be scalable for more complex models in the future.

4 Results

In this section we present the precision, recall, f-measure, and accuracy of Rewind's transcriptions on the following datasets: Nottingham consisting of 1000 or more songs, JSB Chorales consisting of 200 or more songs, Poliner-Ellis consisting of 30 songs, MuseData consisting of 700 songs, the Maps dataset consisting of 169 songs, and a custom dataset that consists of 160 songs split evenly from country, rock, jazz and classical. The custom dataset was added since all of the benchmark datasets currently used in the AMT are currently only classical piano music and orchestral music. All datasets are primarily midi and a synthesizer is used to generate wav except for the Poliner-Ellis and Maps dataset that have a aligned wav file and midi file. Rewind's model ran with two different models and both compared at a 10 ms and 50 ms stride. In Tables 1 and 2, the overall results of Rewind at a 10 ms stride, a standard for AMT systems, at the frame level are demonstrated and compared to Boulanger-Lewandowski's work [6, 24]. The 50 ms results are demonstrated in Table 3, but the results are not reported for the maps dataset. The 10 ms stride results were trained with two parallel GRUs with a linear layer and the 50 ms results were trained with a single GRU and linear layer. The results demonstrated in Table 2 are compared against ConvNet acoustic model at the frame level [24].

Upon examining the table, the Convnet is better overall in accuracy, recall, and f-measure, but Rewind has the higher precision. The ConvNet [24] utilizes a hash beam search to find the most probable sequence. If Rewind was to utilize the same hash beam search, it may have been able to achieve an even better accuracy, recall, and f-measure.

5 Conclusions and Future Work

Rewind demonstrated a encoder-decoder network that is comparable to the results of Boulanger-Lewandowski rnn-rbm [6] in terms of the Nottingham and JSB dataset. It also achieved a higher precision than the rnn-nade [24] on the Maps dataset. However, it suffered from issues in connection with choosing a threshold to generate an on value in the transcription on datasets such as MuseData and the custom dataset built by Rewind. The custom dataset demonstrated that AMT systems can work with multiple genres, but there may be other factors that cause transcription metrics to go down, such as multiple instruments being existent in the song or an improper threshold. Despite these issues, Rewind does manage to follow the underlying frame distribution in the lower classified datasets. Rewind's encoder-decoder has demonstrated a model that has a high precision and comparable results coupled with a web app that can generate transcriptions. Rewind's website provides users with a way to hear and see their transcriptions.

Rewind has demonstrated a model that works at the frame level. Previous work, such as [24], have used a frame level model in conjunction with a note level model to get a note level transcription. One key thing for the encoder-decoder network would be to add another layer, which can do note level transcription and utilize other algorithms from [6] to produce

Table 1: Rewind's results at 10 ms stride for the spectrogram (1 is the proposed model and 2 is the rnn-nade [6])

	Accuracy		Precision		Recall		F-Measure	
Models	1	2	1	2	1	2	1	2
Nottingham	95.1%	97.4%	98.0%		96.9%		97.5%	
JSB	82.8%	91.7%	92.4%		88.8%		90.6%	
Poliner-Ellis	34.4%	79.1%	66.9%		41.5%		34%	
MuseData	34%	66.6%	56.8%		45.9%		50.8%	
Custom	16.2%		51.1%		19.2%		27.9%	

Table 2: Rewind's performance on the Maps dataset compared to [24] at 10 ms.

	Proposed	Simple Auto-Correlation	ConvNet[24]
Accuracy	51.6%	6.4%	**58.87%**
Precision	**76.5%**	21.8%	72.40%
Recall	61.4%	8.2%	**76.50%**
F-Measure	68.1%	11.2%	**74.45%**

Table 3: Rewinds results at a 50 millisecond stride for the spectrogram where 2 is the proposed model and 1 is the Simple Auto-Correlation model

	Accuracy		Precision		Recall		F-Measure	
Models	1	2	1	2	1	2	1	2
Nottingham	21.5%	94.0%	29.2%	97.9%	44.7%	95.9%	35.3%	96.9%
JSB	20.8%	81.6%	32.9%	92.1%	36.2%	87.7%	34.5%	89.9%
MuseData	11.8%	23.0%	15.8%	60.2%	31.9%	27.2%	21.1%	37.4%
Poliner-Ellis	6.6%	42.6%	17.7%	70.5%	9.7%	51.8%	12.5%	55.8%
Custom	8.5%	20.4%	12.2%	44.5%	21.8%	27.3%	15.6%	33.9%

a more probable transcription. This is possible due to the separation of the encoder and decoder in the encoder-decoder network. Another encoder for Rewind could be designed for other problems such as genre classification, audio generation like [26], or audio transformation where a sound is transform into another sound. A deeper architecture could be considered for experimentation for the encoder network, using possibly more GRUs or LSTMs for larger datasets. One other issue that Rewind would like to solve is being able to produce a transcription for each instrument in a song and be able to determine what instrument is being played. Rewind's web has the potential for new features and interfaces for new problems. Rewind could be expanded into an application that allows a user to edit existing music that has been transcribed. Another addition would be to allow Rewind to recognize the lyrics of the music being played. One more thing that Rewind could provide is a way for users to collaborate and learn about music.

Acknowledgement

This material is based in part upon work supported by: The National Science Foundation under grant number(s) IIA-1329469. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

References

[1] James Allwright, *ABC Version of the Nottingham Music Database*, URL: `http://abc.sourceforge.net/NMD/` (visited on 04/10/2016), 2003.

[2] James Allwright, *Bach Choral Harmony Data Set*, URL: `http://archive.ics.uci.edu/ml/datasets/Bach+Choral+Harmony` (visited on 04/10/2016), 2010.

[3] Mert Bay, Andreas F. Ehmann, and J. Stephen Downie, "Evaluation of Multiple-F0 Estimation and Tracking Systems," *Proceedings of the 10th International Society for Music Information Retrieval Conference*, `http://ismir2009.ismir.net/proceedings/PS2-21.pdf`, Kobe, Japan, pp. 315–320, 2009.

[4] Peter J Billam, *MIDI.lua*, URL: `http://www.pjb.com.au/comp/lua/MIDI.html` (visited on 04/10/2016), .

[5] Sebastian Böck and Markus Schedl, "Polyphonic Piano Note Transcription With Recurrent Neural Networks," *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on*, pp. 121–124, DOI: `10.1109/ICASSP.2012.6287832`, 2012.

[6] Nicolas Boulanger-Lewandowski, Yoshua Bengio, and Pascal Vincent, "High-dimensional Sequence Transduction," *2013 IEEE International Conference on Acoustics, Speech and Signal Processing(ICASSP)*, pp. 3178–3182, DOI: `10.1109/ICASSP.2013.6638244`, 2013.

[7] Nicolas Boulanger-Lewandowski, Jasha Droppo, Mike Seltzer, and Dong Yu, "Phone Sequence Modeling With Recurrent Neural Networks," *ICASSP*, IEEE SPS, URL: `http://research.microsoft.com/apps/pubs/default.aspx?id=217321`, 2014.

[8] Chase D. Carthen, "Rewind: A Music Transcription Method," MA thesis, University of Nevada, Reno, 2016.

[9] Chase Carthen, Vinh Le, Richard Kelley, Tomasz Kozubowski, and Frederick C. Harris Jr., "Rewind: A Transcription Method and Website," *Proceedings of the 25th International Conference on Software Engineering and Data Engineering (SEDE 2016)*, Denver, Colorado, USA, pp. 73–78, 2016.

[10] Center for Computer Assisted Research in the Humanities, *MuseData*, URL: `http://musedata.stanford.edu/` (visited on 04/10/2016), 2016.

[11] Kyunghyun Cho, Aaron Courville, and Yoshua Bengio, *Describing Multimedia Content using Attention-based Encoder-Decoder Networks*, eprint: `arXiv:1507.01053`, 2015.

[12] Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio, "Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation," *CoRR* abs/1406.1078 (), URL: `http://arxiv.org/abs/1406.1078`, 2014.

[13] Li Deng, Mike Seltzer, Dong Yu, Alex Acero, Abdel-rahman Mohamed, and Geoff Hinton, "Binary Coding of Speech Spectrograms Using a Deep Auto-encoder," *Interspeech 2010*, International Speech Communication Association, URL: `http://research.microsoft.com/apps/pubs/default.aspx?id=135405`, 2010.

[14] Zhiyao Duan and Emmanouil Benetos, "Automatic Music Transcription," ISMIR, URL: `http://c4dm.eecs.qmul.ac.uk/ismir15-amt-tutorial/`, 2015.

[15] Valentin Emiya, *MAPS Database - A Piano Database For Multipitch Estimation And Automatic Transcription of Music*, URL: `http://www.tsi.telecom-paristech.fr/aao/en/2010/07/08/maps-database-a-piano-database-for-multipitch-estimation-and-automatic-transcription-of-music/` (visited on 04/10/2016), 2008.

[16] Bernd Krueger, *Classical Piano MIDI Page*, URL: `http://www.piano-midi.de/` (visited on 04/10/2016), 2007.

[17] Honglak Lee, Peter Pham, Yan Largman, and Andrew Y. Ng, "Unsupervised Feature Learning For Audio Classification Using Convolutional Deep Belief Networks," *Advances in Neural Information Processing Systems 22*, ed. by Y. Bengio, D. Schuurmans, J. Lafferty, C. Williams, and A. Culotta, pp. 1096–1104, URL: `http://books.nips.cc/papers/files/nips22/NIPS2009_1171.pdf`, 2009.

[18] Nicholas Leonard, Sagar Waghmare, Yang Wang, and Jin-Hwa Kim, *rnn : Recurrent Library for Torch*, eprint: `arXiv:1511.07889`, 2015.

[19] ofoct.com, *Convert WAV (or MP3, OGG, AAC, WMA) to MIDI*, URL: `http://www.ofoct.com/audio-converter/convert-wav-or-mp3-ogg-aac-wma-to-midi.html` (visited on 04/10/2016), 2016.

[20] Graham Poliner, *Automatic Piano Transcription*, URL: `http://labrosa.ee.columbia.edu/projects/piano/` (visited on 04/10/2016), 2008.

[21] Armin Ronacher, *Django The Web Framework For Perfectionists With Deadlines.* URL: `https://www.djangoproject.com/` (visited on 04/10/2016), 2016.

[22] Armin Ronacher, *Flask Web Development, One Drop At a Time*, URL: `http://flask.pocoo.org/` (visited on 04/10/2016), 2016.

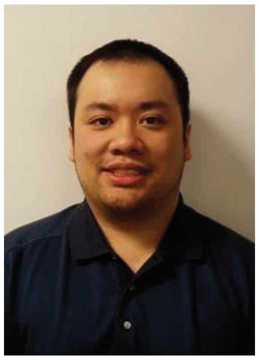
[23] Nicol N. Schraudolph and Terrence J. Sejnowski, “Unsupervised Discrimination of Clustered Data via Optimization of Binary Information Gain,” *Advances in Neural Information Processing Systems*, Morgan Kaufmann, pp. 499–506, 1993.

[24] S. Sigtia, E. Benetos, and S. Dixon, “An End-to-End Neural Network for Polyphonic Piano Music Transcription,” *ArXiv e-prints* (), arXiv: `1508.01774 [stat.ML]`, 2015.

[25] Siddharth Sigtia, Emmanouil Benetos, Srikanth Cherla, Tillman Weyde, Artur S. D’Avila Garcez, and Simon Dixon, “An RNN-Based Music Language Model for Improving Automatic Music Transcription,” *15th International Society for Music Information Retrieval Conference (ISMIR 2014)*, 2014.

[26] Nitish Srivastava, Elman Mansimov, and Ruslan Salakhutdinov, *Unsupervised Learning of Video Representations using LSTMs*, eprint: `arXiv:1502.04681`, 2015.

Chase D. Carthen graduated from the University of Nevada, Reno with both a B.S. and a M.S. in Computer Science and Engineering in 2014 and 2016, respectively. He is currently working in industry as a software engineer. His research interests include human-computer interaction, graphics and simulations, and artificial intelligence.



Vinh Le graduated from the University, Reno with a B.S in Computer Science and Engineering in 2015. Vinh is a Graduate Research Assistant affiliated with the Cyber Infrastructure Lab at the University of Nevada, Reno. He currently aims to earn a Master of Science in Computer Science and Engineering by early 2017 and his research interests consist primarily of Software Engineering, Internet Architecture, and Human-Computer Interaction.

Richard Kelley is currently the chief engineer for the Nevada Advanced Autonomous Systems Innovation Center. He received his BS in Mathematics from the University of Washington in Seattle in 2006, and his MS and PhD in Computer Science and Engineering in 2009 and 2013 respectively. His research interests include robotics, human-robot interaction, machine learning, and unmanned aircraft systems.

Tomasz J. Kozubowski is a Professor in the Department of Mathematics & Statistics at the University of Nevada, Reno. He received his MS in Statistics from the University of Texas, El Paso in 1988 and PhD in Statistics and Applied Probability from University of California, Santa Barbara in 1992. He is a member of the American Statistical Association and the Institute of Mathematical Statistics, and works in the general area of probability and statistics. His research interests include theory of distributions, limit theory for random sums, heavy tailed distributions, extremes, mathematical statistics, financial and insurance mathematics, computational statistics, stochastic models for hydro-climatic phenomena, and fractal scaling processes.

Frederick C. Harris Jr. is currently a Professor in the Department of Computer Science and Engineering and the Director of the High Performance Computation and Visualization Lab and the Brain Computation Lab at the University of Nevada, Reno, USA. He received his BS and MS in Mathematics and Educational Administration from Bob Jones University in 1986 and 1988 respectively, his MS and Ph.D. in Computer Science from Clemson University in 1991 and 1994, respectively. He is a member of ACM (Senior Member), IEEE, and ISCA (Senior Member). His research interests are in parallel computation, computational neuroscience, computer graphics and virtual reality.

Rijndael Algorithm for Database Encryption on a Course Management System[*]

Francis Onodueze[†] and Sharad Sharma[†]
Bowie State University, Bowie, Maryland 20715, USA

Abstract

Effectiveness of any software system depends on techniques employed during access, storage, and retrieval. Securing a course management system with the latest security approaches is vital since it can contain information about students and faculty. Encryption is the most efficient way of securing data stored as it ensures that integrity of data is maintained even if an attacker should gain access to the physical data in the database. Encryption can occur at different levels starting from data, disk to the entire device. This paper presents the implementation of Rijndael Algorithm for a database encryption on a Course Management System, to provide an additional level of security to the information of students, faculty and overall data of the software. We have also provided benefits and drawbacks of various database encryptions based on the amount of data encrypted and te modes of access to keep a balance between efficiency and security. Furthermore, we apply these techniques on a web interface which uses Microsoft technologies to accept users' login details, goes though encryption process, and stores cipher text in the database.

Key Words: Encryption, cipher, database, cryptography, Microsoft dot net.

1 Introduction

Ability of the computer to perform more functions creates the need for more data to be stored. When a computer had less power, it did not store much information because it could be compromised. Only passwords were considered secure data and the computer took as much time to encrypt and decrypt it. However, we have seen the computer's ability increase with proven strength that it can secure more than just passwords. Today, data retrieved from users range from login details to personal data of individuals which they would not share with any human. The confidence users have come from the fact that once their information is encrypted and stored, not even the system admin can retrieve its plan text data. Encryption is a very efficient way of securing data, it helps ensure data confidentiality and integrity in different communication systems, data storage and networks [11]. As defined in [3], encryption algorithms consist of complex mathematical formulas that define the rules of conversion process from plain text to cipher text and vice versa combined with a key. Encryption is achieved using the technique of Cryptography, which is a science that learns the mathematical techniques of keeping information secured [25]. Cryptography converts the original message into unreadable codes and makes sure the original message cannot be retrieved except by reverse process using an appropriate key [7].

Encryption algorithms can come in two forms, public or private Encryption keys, depending on the specifics of each service, application and volume of data to be secured. Public key encryption is a cryptographic system that makes use of pairs of keys. While one key if disseminated publicly, the other is known only to the party that decrypts the message. Amongst the widely known public encryption algorithms is RSA which is a short form for Rivest-Shamir-Adleman who were the developers of the algorithm [18]. Private key encryption is a cryptographic system that uses the same key for encryption and decryption. The cryptographic key used in a symmetric algorithm is often transferred over a secured channel and kept secret by both parties. Some of the private encryption algorithms, also called symmetric algorithms include Data Encryption Standard (DES). Triple DES (TDES), which was derived from encrypting DES three times and Advanced Encryption Standard (AES) which is a standard specification for electronic data. Encryption algorithms can also be classified based on the size of data encrypted in each encryption cycle and size of key. Encryption algorithms are designed to use different length of keys, from 56-bits up to 256-bits, the more the key length, the more secured would be the algorithm and the more resistant it would be for brute force attack [6].

Encryption keys must have two basic attributes to be determined secured; key space and random selection. The key space is determined by the key length and composed of all possible permutations of the keys. Key spaces are designed to make almost impossible for an attacker to search through the set of all possible keys. Random selection determines keys are chosen randomly from all possible keys. Otherwise, an attacker can derive some similar factor that may determine how the key selection was done. Brute force takes the encrypted file and checks all possible combinations of generated keys until a match is found [17]. Most attackers first try dictionary attack before

[*]Extended paper from proceedings of ISCA 25th International Conference on Software Engineering and Data Engineering (SEDE-2016), Marriott Tech Center, Denver, USA, September 26-28, 2016.

[†]Department of Computer Science. Email: onoduezef0108 @students.bowiestate.edu, ssharma@bowiestate.edu.