DeepMnemonic: Password Mnemonic Generation via Deep Attentive Encoder-Decoder Model

Yao Cheng, Chang Xu, Zhen Hai*, and Yingjiu Li

Abstract—Strong passwords are fundamental to the security of password-based user authentication systems. In recent years, much effort has been made to evaluate the password strength or to generate strong passwords. Unfortunately, the usability or memorability of the strong passwords has been largely neglected. In this paper, we aim to bridge the gap between strong password generation and the usability of strong passwords. We propose to automatically generate textual password mnemonics, i.e., natural language sentences, which are intended to help users better memorize passwords. We introduce *DeepMnemonic*, a deep attentive encoder-decoder framework which takes a password as input and then automatically generates a mnemonic sentence for the password. We conduct extensive experiments to evaluate DeepMnemonic on the real-world data sets. The experimental results demonstrate that DeepMnemonic outperforms a well-known baseline for generating semantically meaningful mnemonic sentences. Moreover, the user study further validates that the generated mnemonic sentences by DeepMnemonic are useful in helping users memorize strong passwords.

1 Introduction

Nowadays, user authentication is the key to ensuring the security of user accounts for most online services, such as social media, e-commerce, and online banking. Although various authentication schemes have emerged in recent years, e.g., pattern-based or biometric-based authentication, the password-based authentication remains a prevailing choice in most real-world applications, whose security relies on the difficulty in cracking passwords. Choosing strong passwords becomes extremely important and necessary.

In reality, service providers often present password policies to aid users in creating strong passwords. Such policies may require that *a password be longer than a predefined minimum length, or contain multiple types of characters (letters, numbers, and special characters)*. Such policies are expected to guide the generation of passwords that are resistant to password attacks [1], but users tend to choose passwords that are easy to memorize in practice [2]. As a result, password policies may not be as effective as expected [3].

- *Yao Cheng is with Huawei International, Singapore. E-mail: chengyao101@huawei.com*
- *Chang Xu is with Data61, CSIRO, Australia. E-mail: chang.xu@data61.csiro.au*
- **Corresponding author. Zhen Hai is with Institute for Infocomm Research, A*STAR, Singapore. E-mail: haiz0001@e.ntu.edu.sg*
- *Yingjiu Li is with University of Oregon, Eugene, Oregon, United States. E-mail: yingjiul@uoregon.edu*

Manuscript submitted on 3rd June, 2019.

Much effort has been made to evaluate the strength of passwords [4] [5] [6] or to generate strong passwords [7] [8] [9]. Various methods, on the other hand, have been presented to crack passwords, assess whether passwords are sufficiently strong or not, or reduce password guessability [10] [11] [12]. However, no rigorous effort has been made to address the *memorability* of strong passwords. Strong passwords are usually difficult for users to memorize because their entropy values are beyond many users' memorability [13] [14] [15]. The memorability of strong password has become one of the biggest hindrances to the wide adoption of strong passwords in real-world applications.

One possible approach to addressing this problem is to generate passwords that are not only secure but also easy to remember. Different strategies have been applied to generating word-based memorable passwords, such as pronounceable passwords [16] [17], meaningful passwords [18], and passwords concatenating random words [19]. Unfortunately, some of such strategies were demonstrated to be vulnerable to certain attacks [18] [20], and moreover, there is no theoretical guarantee that the generated passwords are indeed strong. In addition, various user-defined rules were adopted for the generation of expression-based memorable passwords [21] [20] [22]. One limitation of this approach is that the strength of the generated passwords largely depends on the *uncertainty* of the phrases chosen by users. This approach also tends to suffer from the low quality of the generation rules [22].

Instead of generating strong and memorable passwords, recent effort has been made to assist users in remembering passwords using external tools, such as helper cards [23], hint images [24] [25] and engaging games [26]. In particular, Jeyaraman and Topkara [27] proposed a heuristic method that relies on textual hint headlines to deal with the memorability issue of strong passwords. Given a password, they proposed to *search* and *find* an existing natural language headline that suggests the password from a given corpus (i.e., Reuter Corpus Volume 1). If the search is unsuccessful, they would then use an external semantic lexicon named *WordNet* [28] to replace certain words in an existing headline with their synonyms. In this way, they could find a headline or a variant headline and use it as a hint for memorizing a given password. This approach is subject to the following limitations: (i) It can only handle the passwords composed of alphabetic characters. Manual intervention is needed to tackle digits or special characters, which are non-trivial

for the composition of strong passwords in reality. (ii) Meaningful hint headlines are often missing by simply searching the given corpus while creating variant headlines at the word level by using an external lexicon may result in syntactically incorrect or semantically inconsistent headline sentences. (iii) It only works for short passwords that consist of 6 or 7 characters due to the limited lengths of the headlines in the corpus.

In this work, given strong passwords, we propose to automatically generate *mnemonics*, i.e., *natural language* sentences, to bridge the gap between security and memorability of the passwords. Specifically, we introduce a new mnemonic generation system named *DeepMnemonic*, which is capable of generating a *semantically meaningful* mnemonic sentence for each given textual password. The core of DeepMnemonic is an attentive encoder-decoder neural network. It learns to transform any given user password into a mnemonic sentence by first encoding the password character sequence and then decoding the encoded information into the mnemonic sentence via a specially designed attention strategy. Both encoder and decoder are implemented with recurrent neural networks that can capture the contextual dependency among pairwise input passwords and mnemonic sentences. The key insight of DeepMnemonic is inspired by a cognitive psychology theory [29], which states that the human ability to memorize and recall certain information is positively influenced by associating additional semantic content with the information [27]. The natural language mnemonic sentences generated by DeepMnemonic can provide such semantically meaningful content for users to remember and recall the given strong passwords.

Different from the existing textual password hint systems [27], DeepMnemonic enjoys several unique properties: (i) Feature-free: DeepMnemonic is an end-to-end solution, which aims to directly map a given textual password to its corresponding mnemonic sentence, and no manual feature engineering is needed in the learning process. (ii) Long-password-friendly: The attention-based recurrent neural network component in DeepMnemonic is capable of identifying salient information in long passwords for generating semantically meaningful sentences. (iii) Learning-adaptive: The learning of DeepMnemonic is fitted to different types of password-sentence training pairs so as to process diversified passwords and meet various mnemonic generation requirements.

In summary, we have made the following main contributions in this work:

- We introduce DeepMnemonic, a mnemonic sentence generation system, to help users remember strong passwords. DeepMnemonic utilizes an encoder-decoder neural network model to generate meaningful mnemonic sentences for given strong passwords.
- We quantitatively evaluate the capability of DeepMnemonic in mnemonic sentence generation. Our experimental results show that DeepMnemonic achieves 99.09% MP (Mnemonic Proportion) and 16.47 BLEU-4 (BiLingual Evaluation Understudy), outperforming an *n-gram* language model baseline (83.62% MP and 5.09 BLEU-4).
- We conduct a user study to qualitatively evaluate the helpfulness of DeepMnemonic. The results demonstrate that DeepMnemonic helps users with 54.47% decrease in time spent on remembering passwords and 57.14% decrease in recall error measured by the edit distance between each pair of the given password and its recalled version.

The rest of this paper is organized as follows. Section 2 describes the password mnemonic generation problem and its application scenario. Section 3 introduces DeepMnemonic, a deep attentive encoder-decoder system that can generate mnemonic sentences for the given textual passwords. In Section 4 and Section 5, we evaluate DeepMnemonic in experiment and user study, respectively. Section 6 discusses several usability issues of DeepMnemonic. Section 7 summarizes the related work, and Section 8 concludes this paper.

2 Problem Statement and Application Scenario

Given a strong textual password that often consists of random characters, the aim of this work is to assist common users in remembering the password properly. According to a famous cognitive psychology theory [29], in order to memorize a piece of information, it is helpful to associate some external tools or contextual contents with the information. More specifically, the cognitive psychology theory [29] explains that the information to be remembered can be a list of items, for example, a list of characters in a password. For the semantic content, it is not limited to a vivid image representing, symbolizing, or suggesting each item of semantic information. It indicates that, in order to better memorize a piece of information, users can construct such semantic content and associate it with the information. Inspired by the theory, we propose DeepMnemonic, an intelligent system that helps users construct the semantic content by suggesting a meaningful mnemonic sentence corresponding to each character in the password. For example, given a text password *Tcahm,"Wac?"* to be remembered, a semantic content can be *The child asked her mother," Who are children?"*, where the first letter of each word in the mnemonic sentence is suggesting and corresponding to each character in the password.

Technically, the problem of password mnemonic generation is closely related to the machine translation problem in natural language processing. Both tasks aim to address a sequence-to-sequence learning problem. Specifically, in machine translation, the goal is to generate a translated sentence in a target language given a sentence in a source language, whereas in our case, we focus on generating a mnemonic sentence for a given password (which is a sequence of characters). We propose to exploit a neural sequence-to-sequence language model [30] to *translate* a given password (a character sequence) into a meaningful mnemonic sentence (a word sequence).

Formally, let D be a set of N pairs of password and mnemonic sentence (X_n, Y_n) ($n \in \{1, \cdots, N\}$), where X_n denotes a password of T characters, $X_n = x_1, x_2, \cdots, x_T$, where the character x_t ($t \in \{1, \cdots, T\}$) can be alphabetic, punctuation, digital, and special characters, while Y_n refers to a mnemonic sentence of T dictionary terms, $Y_n = y_1, y_2, \cdots, y_T$, where the term y_t ($t \in \{1, \cdots, T\}$) can be words, punctuation marks, numbers, and special tokens. In the following, we use *words* or *tokens* to refer to dictionary terms. For each pair of password and mnemonic sentence, a

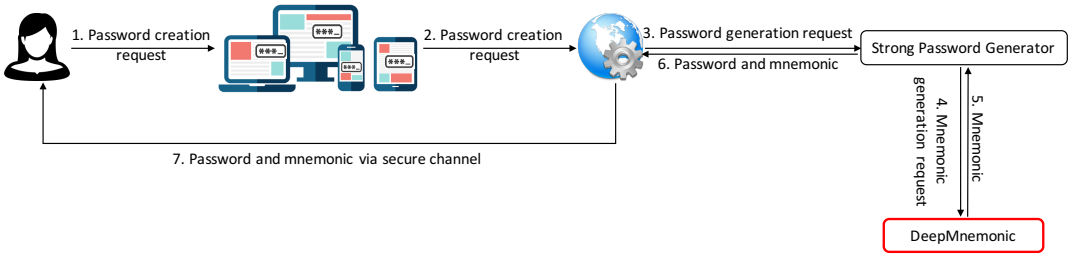


Fig. 1: An applicable scenario of DeepMnemonic in the password-based authentication system.

one-to-one mapping relationship exists between the characters of the password and the words of the mnemonic sentence, where each character of the password may appear at certain position of the corresponding mnemonic word. For example, given a text password *O,y,slt.*, a mnemonic sentence can be *Oh, yes, something like that.*, where each alphabetic character of the password corresponds to the first letter of respective words in the mnemonic sentence.

We formulate the mnemonic sentence generation as a natural language generation problem. Given a (strong) password X_n, the problem is to automatically generate a semantically coherent and meaningful mnemonic sentence Y_n of the same length as X_n. Our solution, named *DeepMnemonic*, employs a neural attentive encoder-decoder language model [30] to generate textual mnemonic sentences for passwords. DeepMnemonic builds on a sequence-to-sequence learning framework, which encodes an input password (i.e., a sequence of characters) and automatically generates a mnemonic sentence (i.e., a sequence of words) based on the encoded contextual information.

Figure 1 shows an applicable scenario of DeepMnemonic in the conventional password-based authentication system. When a user requires to create a password for registration (step 1 and step 2), the authentication service requests the password generator to generate a strong password (step 3). Typically, the generated password may not be easy for the user to remember due to the randomness of the password [13]. To improve the memorability of the strong password, the password generator requests DeepMnemonic to produce a mnemonic sentence corresponding to the generated password (step 4). Once the password generator receives the response from DeepMnemonic (step 5), it returns the generated password as well as the corresponding mnemonic sentence to the service (step 6). Finally, the service distributes the password and mnemonic information to the user via a secure channel (step 7).

Notice that DeepMnemonic does not modify the strong passwords generated by the password generator, and hence does not compromise the security of the generated passwords, under the assumption that users keep both passwords and associated mnemonic sentences secure. The passwords' resilience to existing password attacks, such as brute force attacks, dictionary attacks, and guessing attacks, relies on strong password generation which has been well studied independently of our research. DeepMnemonic does not generate strong passwords but focuses on the usability of strong passwords. It can generate a corresponding mnemonic sentence for the password so as to overcome the hindrance to the practical use of strong password generation techniques.

3 Methodology

3.1 Overview

DeepMnemonic applies a deep attentive encoder-decoder learning model [30] to learn a sequence-to-sequence mapping from an input password (i.e., a character sequence) to an output mnemonic sentence (i.e., a word sequence). The encoder of DeepMnemonic captures the underlying contextual *"meaning"* of a password from its sequence of characters, while the decoder generates a corresponding mnemonic sentence based on the encoded content of password. Typically, the lengths of the given passwords are variable, and strong passwords may consist of long sequences of random characters. DeepMnemonic may lose focus in its process, if it treats all characters of an input password equally. To tackle this issue, DeepMnemonic exploits an attention mechanism [30] to dynamically determine which parts of an input password are more relevant to generating a semantically meaningful mnemonic sentence.

Figure 2 shows the encoder-decoder learning framework of DeepMnemonic. The encoder first takes as input a textual password X_n of length T,

$$X_n = (x_1, x_2, \cdots, x_t, \cdots, x_T),$$

where $x_t \in \mathbb{R}^{V_C}, t \in \{1, \cdots, T\}$ denotes a character, and V_C is the size of the input *character vocabulary*. It derives a hidden context-aware summary or "meaning" of the sequence of characters from input password via a bidirectional recurrent neural network (BiRNN) [31]. Based on the encoded hidden "meaning" of the password, the decoder of DeepMnemonic utilizes an attentive mechanism [30] to generate the corresponding mnemonic sentence Y_n word by word,

$$Y_n = (y_1, y_2, \cdots, y_t, \cdots, y_T),$$

where $y_t \in \mathbb{R}^{V_W}$ denotes a word, and V_W is the size of the output *word vocabulary*.

3.2 Encoder

Given an input password X_n, we employ a BiRNN to derive the hidden semantic representation as the meaning of the password. BiRNN consists of both forward and backward processes, where the forward process reads the password character sequence in the original order, while the backward process reads the sequence in the reverse order. BiRNN can thus capture contextual patterns in both directions (left and right) for each password character by summarizing the information not only from the preceding characters but also from the following ones in the sequence.

Formally, by using the gated recurrent units (GRU) [32], which is a popular choice for modern RNNs, the forward

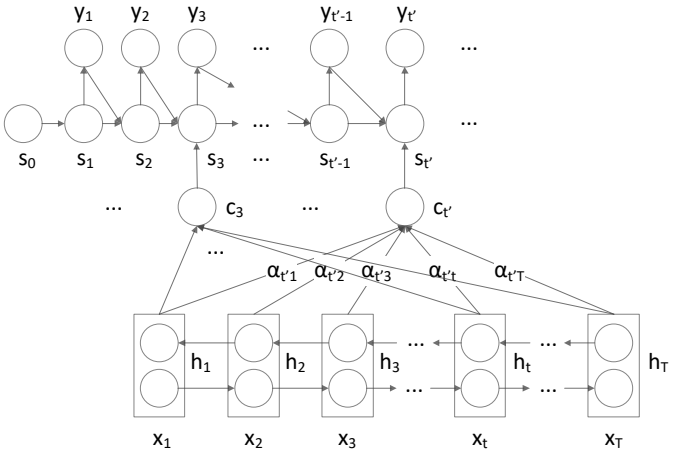


Fig. 2: The encoder-decoder learning framework of DeepMnemonic.

process of BiRNN computes the hidden state h_t at a given time step t as follows. First, the forward reset gate $\vec{r}_t$ is computed as,

$$\vec{r}_t = \sigma(\vec{W}^{(r)} E^{(C)} x_t + \vec{U}^{(r)} \vec{h}_{t-1}), \tag{1}$$

where σ is the sigmoid function, $E^{(C)} \in \mathbb{R}^{m \times V_C}$ is the character embedding matrix for the password characters, $\vec{W}^{(r)} \in \mathbb{R}^{n \times m}$ and $\vec{U}^{(r)} \in \mathbb{R}^{n \times n}$ are trainable parameter matrices, and m and n refer to the dimensionalities of the character embedding and the hidden state vector, respectively.

Similarly, the forward update gate $\vec{z}_t$ is computed as,

$$\vec{z}_t = \sigma(\vec{W}^{(z)} E^{(C)} x_t + \vec{U}^{(z)} \vec{h}_{t-1}), \tag{2}$$

where $\vec{W}^{(z)} \in \mathbb{R}^{n \times m}$ and $\vec{U}^{(z)} \in \mathbb{R}^{n \times n}$ are trainable parameter matrices.

Then, the forward hidden state vector $\vec{h}_t$ is computed as,

$$\vec{h}_t = (1 - \vec{z}_t) \circ \vec{h}_{t-1} + \vec{z}_t \circ \vec{\tilde{h}}_t, \tag{3}$$

where

$$\vec{\tilde{h}}_t = \tanh(\vec{W} E^{(C)} x_t + \vec{U}[\vec{r}_t \circ \vec{h}_{t-1}]),$$

both $\vec{W} \in \mathbb{R}^{n \times m}$ and $\vec{U} \in \mathbb{R}^{n \times n}$ are trainable parameters, and $\circ$ refers to an element-wise multiplication.

Following the aforementioned steps, the backward hidden state $\overleftarrow{h}_t$ can be computed reversely for any given time step t. Note that the character embedding matrix $E^{(C)}$ is shared for both forward and backward processes of BiRNN. Next, we concatenate both forward and backward hidden states $\vec{h}_t$ and $\overleftarrow{h}_t$ for each time step t, and derive a set of overall hidden states $(h_1, \cdots, h_T)$, where

$$h_t = \begin{bmatrix} \vec{h}_t \\ \overleftarrow{h}_t \end{bmatrix}. \tag{4}$$

3.3 Decoder

Based on the encoded hidden states $(h_1, \cdots, h_T)$ of the input password, the decoder of DeepMnemonic employs an attention mechanism to evaluate the importance of individual hidden states. Then, it derives an overall context-aware representation of the hidden states in order to generate each target word in the corresponding mnemonic sentence.

In particular, for each encoded hidden state h_t at time step t, the decoder first computes an attentive weight $\alpha_{t',t}$ with regard to the contextual state $s_{t'-1}$ at time step t',

$$\alpha_{t',t} = \frac{\exp(e_{t',t})}{\sum_{\tau=1}^{T} \exp(e_{t',\tau})}, \tag{5}$$

where

$$e_{t',t} = {v^{(a)}}^T \tanh(W^{(a)} s_{t'-1} + U^{(a)} h_t)$$

is an alignment model that evaluates the relevance between the input hidden state h_t and the previous output hidden state $s_{t'-1}$. The $v^{(a)} \in \mathbb{R}^{n'}$, $W^{(a)} \in \mathbb{R}^{n' \times n}$, and $U^{(a)} \in \mathbb{R}^{n' \times 2n}$ are trainable parameters, and n' refers to the dimensionality of the hidden state vector of the attention layer.

Then, the attentive context vector c_t is computed via a weighted sum,

$$c_{t'} = \sum_{t=1}^{T} \alpha_{t',t} h_t. \tag{6}$$

We can then compute the hidden state $s_{t'}$ of the decoder at time t' given the hidden states of encoder via a decoder GRU as below,

$$s_{t'} = (1 - z_{t'}) \circ s_{t'-1} + z_{t'} \circ \tilde{s}_{t'}, \tag{7}$$

where

$$\begin{aligned} \tilde{s}_{t'} &= \tanh(W E^{(W)} y_{t'-1} + U[r_{t'} \circ s_{t'-1}] + C c_{t'}), \\ z_{t'} &= \sigma(W^{(z)} E^{(W)} y_{t'-1} + U^{(z)} s_{t'-1} + C^{(z)} c_{t'}), \\ r_{t'} &= \sigma(W^{(r)} E^{(W)} y_{t'-1} + U^{(r)} s_{t'-1} + C^{(r)} c_{t'}). \end{aligned} \tag{8}$$

In the above formulas, $E^{(W)} \in \mathbb{R}^{m \times V_W}$ is the word embedding matrix for the output mnemonic sentence; $W, W^{(z)}, W^{(r)} \in \mathbb{R}^{n \times m}$, $U, U^{(z)}, U^{(r)} \in \mathbb{R}^{n \times n}$ and $C, C^{(z)}, C^{(r)} \in \mathbb{R}^{n \times 2n}$ are all trainable parameters. m and n are the dimensionalities of the word embedding and the hidden state vector, respectively. Note that the initial hidden state of decoder s_0 is computed as,

$$s_0 = \tanh(W^{(s)} \overleftarrow{h}_1), \tag{9}$$

where $W^{(s)} \in \mathbb{R}^{n \times n}$ is the parameter matrix.

Next, given the decoder state $s_{t'}$, the attentive context vector $c_{t'}$, and the previous generated word $y_{t'-1}$, we follow [30] and define the probability for generating the target word $y_{t'}$ as follows,

$$p(y_{t'} | s_{t'}, c_{t'}, y_{t'-1}) \propto \exp(y_{t'-1}^T W^{(o)} l_{t'}), \tag{10}$$

where

$$l_{t'} = [\max\{\tilde{l}_{t',2k-1}, \tilde{l}_{t',2k}\}]_{k=1,\cdots,K}^T,$$

and $W^{(o)} \in \mathbb{R}^{V_W \times K}$. The hidden state $\tilde{l}_{t'}$ is computed as follows,

$$\tilde{l}_{t'} = U^{(o)} s_{t'} + V^{(o)} E^{(W)} y_{t'-1} + C^{(o)} c_{t'}, \tag{11}$$

where $U^{(o)} \in \mathbb{R}^{2K \times n}$, $V^{(o)} \in \mathbb{R}^{2K \times m}$, and $C^{(o)} \in \mathbb{R}^{2K \times 2n}$ are trainable parameters.

4 EXPERIMENTS

We conducted quantitative experiments on a large-scale publicly available dataset to evaluate DeepMnemonic for mnemonic sentence generation. We also carried out a case study to explore the influence of password lengths and digital/special characters. In addition, we provided a visualized understanding of the attention mechanism in DeepMnemonic.

4.1 Dataset

The "Webis-Simple-Sentences-17" is a publicly available data source for analyzing *expression-based passwords*, and the sentences in the dataset are similar to the human-chosen mnemonics in terms of syllable distribution [22]. After filtering out those sentences that contain non-English words, we derived a password from each remaining sentence by concatenating the first letter of each word and the rest special characters (including punctuation marks and numerical digits) in the original order as shown in the sentence. For example, a password *O,y,slt.* can be derived from the following sentence, *Oh, yes, something like that.* In this way, we collected totally 500,000 pairs of passwords and sentences, forming a ground-truth dataset for building DeepMnemonic. It is worth noting that, in creating the ground truth data, it is also possible to derive a password by concatenating the last (or a middle) letter of each word and the rest special characters in each sentence.

In preprocessing the ground-truth data, all words of each sentence were converted to lowercase, which was helpful to reduce the size of vocabulary for the language generation. In particular, by following the given passwords, we can easily restore the corresponding words of the generated mnemonic sentences to uppercase wherever applicable. In addition, extra symbols, such as <s> and </s>, were inserted at the start and the end of each sentence to indicate its boundary. A special *unknown* symbol, i.e., <UNK>, was included in the vocabulary, and would be used for the generation of sentences when no appropriate words can be predicted. In the pairwise ground-truth dataset, the minimum and maximum lengths of passwords are 8 and 16, respectively, which are compatible with the password length requirements in many authentication systems.

Among the ground truth dataset, we randomly selected 450,000 pairs as training data, and used the rest 50,000 for testing. Among the training data, 20% of the pairs were randomly held out as validation data for model selection. After preprocessing, we obtained a vocabulary of 109,584 words (tokens) for the mnemonic sentence generation task.

4.2 Experimental Setting

In DeepMnemonic, the hidden layer sizes of both encoder and decoder were set to 256, which were optimally selected using the validation set. The dropout strategy has been previously shown effective to prevent neural network models from overfitting [33] and the dropout rate was set to 0.2 in our experiments.

Note that no extra information, e.g., pre-defined generation rules, is required beyond training data. Once the training process is done, the DeepMnemonic can be used to automatically generate a mnemonic sentence for any given password.

4.2.1 Beam Search

Simply generating the *best* word that achieves the highest predictive probability at each time step may not always result in an overall semantically meaningful sentence in practice. Therefore, a left-to-right beam search strategy [34] was employed to find the most likely mnemonic sentence [35] [30] for each given password.

In particular, the beam search based decoder stores a predetermined number b (beam width) of partial sentences, where each partial sentence is a prefix of a candidate mnemonic. At each time step, each partial sentence in the beam grows with a possible mnemonic word from the decoder vocabulary. Clearly, this process would greatly increase the number of candidate sentences. To overcome this issue, only b most likely candidates are maintained in terms of their predicted probabilities. Once the end of sentence symbol is appended to a candidate mnemonic, it is included in the set of full mnemonic sentences. In general, the wider the beam width b is, the more candidate mnemonics the decoder searches for, and thus the better results could be achieved.

4.2.2 Baseline

The n-gram language model is widely known as one of the dominant methods for probabilistic language modeling [36]. As a non-parametric learning method, it primarily utilizes the preceding sequence of $n-1$ words to estimate a conditional probability for the prediction of the current word. Various values of n were tested for n-gram, and the *bigram* language model ($n=2$) achieved decent generation performance and was chosen as the baseline to benchmark the proposed DeepMnemonic.

The bigram language model depends on the preceding word for estimating the conditional probability, and then generates the current word that has the highest probability. Therefore, the bigram model is unable to automatically figure out the relationship between a given password and its mnemonic sentence, i.e., the correspondence between the alphabetic characters of the password and the first letters of mnemonic words in the sentence. To properly apply the bigram language model to password mnemonic generation, we manually adopted a pre-defined generation rule. Specifically, to generate a mnemonic word, it is required that the word not only achieve the highest conditional probability (given its preceding word), but its first letter also must be identical to the corresponding character in the given password.

4.2.3 Evaluation Metrics

Two metrics were used to evaluate the quality of generated mnemonic sentences via DeepMnemonic and the baseline method. One metric is BLEU (BiLingual Evaluation Understudy), which is one of the most popular metrics for evaluating the quality of machine translation task in natural language processing [37]. Following [30], we used BLEU to evaluate the quality of the generated mnemonic sentences with respect to ground-truth sentences in the test set, where the quality refers to the correspondence between each pair of the generated sentence and the ground-truth sentence. In other words, the closer the generated mnemonic sentence is to the ground-truth sentence, the more meaningful and consistent it is. BLEU-n is defined as follows (the higher, the better).

$$BLEU\text{-}n = B \cdot exp(\frac{1}{N'}\sum_{n=1}^{N'} logp_n),$$

where N' is the maximum length of n-grams and $N' = 4$ was used in the experiments. B refers to brevity penalty, and is computed as

$$B = \left\{ \begin{array}{ll} 1 & if \quad c > r \\ e^{1-r/c} & if \quad c \leq r \end{array} \right\},$$

where c is the size of the generated mnemonic set, and r is the size of the ground truth sentence set. The modified n-gram precision p_n is computed as

$$p_n = \frac{\sum_{C \in \mathcal{H}} \sum_{\text{n-gram} \in C} Count_{clip}(\text{n-gram})}{\sum_{C' \in \mathcal{H}} \sum_{\text{n-gram}' \in C'} Count(\text{n-gram}')},$$

where $\mathcal{H}$ is the set of generated mnemonic sentences, $Count(\text{n-gram}')$ is the number of n-gram's in a generated mnemonic, and $Count_{clip}(\text{n-gram})$ is the clipped number of the n-gram of a generated mnemonic with regard to the corresponding ground-truth mnemonic sentence.

The other metric, Mnemonic Proportion (MP), is specifically defined for testing the matching proportion between pair-wise passwords and mnemonic sentences. Specifically, given each pair of password X_n and generated mnemonic sentence Y_n, MP calculates the proportion of the cases where each word y_t of mnemonic Y_n does start with the corresponding character x_t in password X_n, as shown below:

$$MP = \frac{\sum_{n=1}^{N} MP_n}{N},$$

where N is the size of the test set. Then MP_n is defined as,

$$MP_n = \frac{|\{y_t | \, y_t \in Y_n \text{ and } FirstLetter(y_t) = x_t\}|}{|X_n|}$$

where $|X_n|$ is the length of password X_n, and $FirstLetter(\cdot)$ is a function that returns the first letter of a word.

4.3 Experimental Results

This section reports the results of mnemonic sentence generation via DeepMnemonic and the baseline model in terms of MP and BLEU. DeepMnemonic ran on Nvidia Tesla P100 GPU with 16GB memory. Its training process with 450,000 training data pairs took around 15 hours, and the inference on the entire test data set with 50,000 passwords took about 5 minutes.

4.3.1 MP

Table 1 shows the MP results of the DeepMnemonic and bigram language model (the higher, the better). As we can see, DeepMnemonic achieves much better results compared to Bigram over different beam widths.

Given beam width $b = 1$, DeepMnemonic attains an MP value of 99.09%, while Bigram only achieves 83.62% MP. When the beam width increases to 5, the Bigram MP increases up to 98.44%, but is still lower than DeepMnemonic. Surprisingly, increasing the beam width does not lead to significant gain to DeepMnemonic. This suggests that DeepMnemonic can achieve a high MP value given a small beam width $b = 1$, and it is not sensitive to the choice of width of the beam search.

TABLE 1: The MP results of DeepMnemonic and bigram language model (Bigram) given beam width (b) of 1 and 5.

	DeepMnemonic	Bigram
$b = 1$	99.09%	83.62%
$b = 5$	99.15%	98.44%

4.3.2 BLEU

This section reports the BLEU-n scores for the generated sentences with regard to ground-truth mnemonic sentences [37] [35]. Table 2 lists the BLEU-n results with the order $n \in \{1, 2, 3, 4\}$ ($N' = 4$) for DeepMnemonic and Bigram given different beam width $b \in \{1, 5\}$. Overall, DeepMnemonic outperforms Bigram significantly and consistently in all cases.

If beam width is fixed at $b = 1$ or $b = 5$, the BLEU-n score for each model decreases as the order n grows from 1 to 4. This is expected, as increasing the order n results in a stricter evaluation of BLEU for the generated sentences, which is consistent with the findings in [37]. In other words, not only are the words between each pair of the generated and the ground-truth sentences required to be identical, but the order of the words in the n-gram also needs to be exactly the same.

DeepMnemonic achieves the best BLEU-1 of 45.29 compared to the baseline Bigram given the beam width $b = 5$. Roughly, this means that about 45 out of 100 generated mnemonic words (unigrams) are identically matched to the ground truth. In addition, the results again show that DeepMnemonic is robust to the width of the beam search. In contrast, the BLEU-n values (from 1 to 4) of Bigram improve significantly by increasing the beam width from $b = 1$ to $b = 5$. But the best BLEU-1 (37.22) of Bigram at $b = 5$ is still lower than DeepMnemonic at $b = 1$.

In addition, it is worth noting that BLEU-n is designed to measure the *correspondence* between each pair of the generated mnemonic sentence and ground-truth sentence. However, it is possible that a generated mnemonic sentence is helpful to assist users in memorizing a given password, but it may have a low BLEU-n score if the generated sentence does not match to its ground truth very well. To mitigate this issue, Section 5 further conducts a user study about the helpfulness of DeepMnemonic, where participants are invited to memorize and recall the assigned passwords using mnemonic sentences generated by DeepMnemonic.

TABLE 2: The BLEU scores of DeepMnemonic and bigram language model (Bigram) given beam width (b) 1 and 5.

	DeepMnemonic		Bigram	
	$b = 1$	$b = 5$	$b = 1$	$b = 5$
BLEU-1	45.17	45.29	28.82	37.22
BLEU-2	30.26	30.38	14.72	22.07
BLEU-3	22.33	22.44	8.49	14.29
BLEU-4	16.47	16.57	5.09	9.48

4.4 Case Study

In this section, we conduct a case study to provide a complementary understanding of the generated mnemonic

sentences. The case study reveals the influence of password lengths and digital/special characters on mnemonic generation. Moreover, we visualize the effectiveness of the attention mechanism in DeepMnemonic to explain its semantic meaningfulness.

Given a list of 10 randomly selected passwords of different lengths, Table 3 shows the corresponding mnemonic sentences generated via DeepMnemonic and Bigram methods. Note that the row "Original" shows the ground-truth sentences from our ground-truth dataset.

4.4.1 Password Length

It can be observed that, for short passwords, all the sentences generated by either DeepMnemonic or Bigram match all the password characters (i.e., 100% MP). However, as the password length grows, the unknown token "<UNK>" begins to appear more frequently in the mnemonic sentences generated by Bigram compared to DeepMnemonic. The main reason is that, when generating a sentence, it is sometimes difficult for the Bigram model to find a word that not only identically matches the given password character but also has the conditional probability greater than zero given the preceding word. For example, Bigram begins to generate the unknown token "<UNK>" from the third position onwards in Case 7. Given the first generated word "but", the model identifies the next word "z2", which has the highest probability and also starts with the character "z" of the given password. However, when continuing to generate the next word based on "z2", Bigram fails to find any words that start with the password character "e" and also have the positive conditional probability. As a result, Bigram generates an unknown token instead. In contrast, DeepMnemonic does not have such issue, and for each given password, it can complete the automatic generation of an entire meaningful sentence.

Figure 3 plots the impact of password lengths on MP values at $b = 5$. We can observe that for both Bigram and DeepMnemonic, MP values decrease as the passwords become longer. The longer the passwords are, the more difficult the task of generating semantically sensible sentences is. It is clear that the MP values of DeepMnemonic are always better than those of Bigram at different password lengths, suggesting that DeepMnemonic generates better mnemonic sentences from the given passwords. For example, the mnemonic sentence generated by DeepMnemonic in Case 9 in Table 3 is more sensible and memorable than that by Bigram, although both sentences match all characters of the given password.

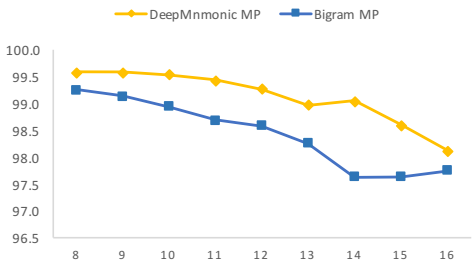


Fig. 3: Impact of the password length on MP values (%) given $b = 5$.

4.4.2 Digital and Special Characters

In many password generation systems, digital and special characters play an important role in creating strong passwords. DeepMnemonic is able to handle such characters when generating mnemonic sentences for strong passwords.

As shown in Case 3 of Table 3, DeepMnemonic generates a digital sequence "1937" from number "1" in the given password, and then it concatenates previously generated words to form a meaningful expression, "in february 1937", for the given password subsequence "iF1". In contrast, Bigram generates the words "it for 1" for the same subsequence, which is less readable and not much helpful for remembering the password. Overall the mnemonic sentence generated by Bigram is not as meaningful as the sentence by DeepMnemonic. In Case 6, DeepMnemonic generates a sensible mnemonic phrase "75 causalities in 24 hours" from a password segment "7ci2h", and also produces an overall semantically meaningful mnemonic sentence. Although Bigram generates a meaningful phrase "70 countries in 2006 have" from the same password segment, unfortunately, it ends up with a grammatically incorrect and semantically inconsistent sentence.

In general, it is challenging to handle special characters of passwords such as "/" and "*" during generating mnemonic sentences. For example, in Case 2, given a slash character "/" between "a" and "e" of the password, DeepMnemonic generates a sensible phrase " author / editor", where "/" normally represents "or", while the Bigram model outputs a strange phrase "a / etc" for the same password segment. In Case 4 of Table 3, following the password segment "*g*", DeepMnemonic generates a reasonable phrase "*good*". Though it is not identical to the original one ("*giddy*"), semantically, this phrase can be used to highlight the meaning of "good" in the generated sentence. However, Bigram fails to generate either a short sensible phrase or an entire semantically meaningful sentence.

One more interesting example is the colon symbol ":" in Case 10, which is typically used to indicate the start of an utterance. Surprisingly, DeepMnemonic generates a name "alan" (Alan) for the character "A" in front of the colon ":", and then generates an utterance for the subsequence of characters following the colon symbol. Bigram fails again to handle this special character.

4.4.3 Attentive Generation of Mnemonic Sentences

Generally, a mnemonic sentence, which not only literally covers the characters of the given password, but is also semantically meaningful, is more useful for users to memorize every password character properly.

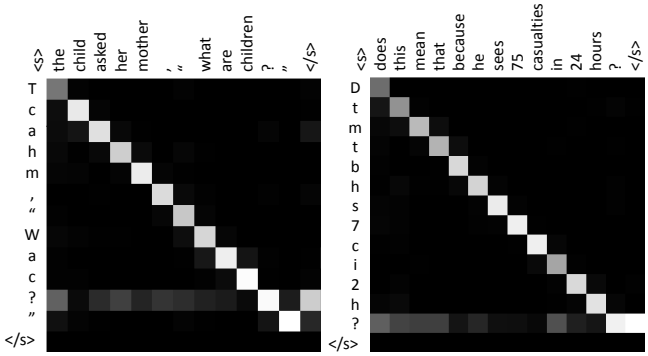
As shown in Case 8 of Table 3, DeepMnemonic generates a mnemonic sentence, "today, i would like to read the letter of karl james." from password "T,IwltrtloKJ." Although this is not identical to the ground-truth sentence, i.e., "today, i would like to recognize the life of ken jablonski.", our generated mnemonic sentence seems easier for users to follow and recall the corresponding password. DeepMnemonic decodes the given password segment "KJ" as a name, "karl james", and surprisingly, it turns out to be a new person name that does not even appear in the training data. This demonstrates the ability of DeepMnemonic to capture the semantic context of input passwords as well as the relationship between each pair of password and mnemonic sentence. This also shows that the attention mechanism in

TABLE 3: Randomly selected passwords and the corresponding mnemonic sentences generated by DeepMnemonic and Bigram (*Original* means that the mnemonic sentences come from the ground-truth dataset).

No.	Length	Items	Values
Case 1	Short	Password	Y m s , b t o .
		Original	you'll miss some , but that's okay .
		DeepMnemonic	you may say , but that's ok .
		Bigram	you might say , but the other .
Case 2	Short	Password	S I b a a / e ?
		Original	should i buy a acoustic / electric ?
		DeepMnemonic	should i be an author / editor ?
		Bigram	so i bought at a / etc ?
Case 3	Short	Password	T f i d i F 1 .
		Original	this finding is diagrammed in figure 1 .
		DeepMnemonic	the festival is due in february 1937 .
		Bigram	the first i do it for 1 .
Case 4	Medium	Password	B I j c d t * g * t .
		Original	but i just can't do the * giddy * thing .
		DeepMnemonic	but i just can't do the * good * thing .
		Bigram	but i just can't do they * go <UNK> <UNK> <UNK>
Case 5	Medium	Password	T c a h m , " W a c ? "
		Original	the child asks his mother , " what are circles ? "
		DeepMnemonic	the child asked her mother , " who are children ? "
		Bigram	they can also have more , " we are called ? "
Case 6	Medium	Password	D t m t b h s 7 c i 2 h ?
		Original	does this mean the book has sold 7 copies in 24 hours ?
		DeepMnemonic	does this mean that because he sees 75 casualties in 24 hours ?
		Bigram	during the more than before he said 70 countries in 2006 have ?
Case 7	Medium	Password	B Z e p a l i h h t c h .
		Original	but zeus ever pursued and longed in his heart to catch her .
		DeepMnemonic	but zelotes ever pushed a line in his head to crucify him .
		Bigram	but z2 <UNK> <UNK> <UNK> <UNK> <UNK> <UNK> <UNK> <UNK> <UNK> <UNK> <UNK>
Case 8	Medium	Password	T , I w l t r t l o K J .
		Original	today , i would like to recognize the life of ken jablonski .
		DeepMnemonic	today , i would like to read the letter of karl james .
		Bigram	this , it was like to read the lord of knowledge <UNK> <UNK>
Case 9	Long	Password	D t s w i N - D , l a t a p .
		Original	during the six weekends in new - delhi , lunch and tea are provided .
		DeepMnemonic	during the second week in november - december , lakes awoke to a pond .
		Bigram	during the same way is no - day , look at the air pollution .
Case 10	Long	Password	A : Y , b t e n t b a i a l .
		Original	ac : yes , but the egg needs to be altered in a lab.
		DeepMnemonic	alan : yeah , but that's exactly not the best album in a league .
		Bigram	a : yes , but the early next to be an important as long .

DeepMnemonic captures not only the full view of an input password context but also its salient parts, both of which are useful for mnemonic word inference at each generation step.

Figure 4 visualizes the attention weights in alignment between each pair of input password (y-axis) and generated mnemonic sentence (x-axis) for Case 5 and Case 6. Each pixel denotes the attention weight $\alpha_{t',t}$ of the t-th password character with regard to t'-th target mnemonic word in grayscale (0: black, 1: white). The brighter the pixel is, the more important the password character is to the generation of the corresponding mnemonic word. From the heat map, we can observe that DeepMnemonic concentrates on the important characters of an input password when it decodes individual target mnemonic words in the generation phase. In addition, the attention layer also takes into account the contextual neighboring characters of each password for mnemonic sentence generation. Thanks to the attention mechanism, DeepMnemonic automatically learns the alignment and discovers which characters in the input password are more important for generating a semantically meaningful mnemonic sentence.



(a) Case 5. (b) Case 6.

Fig. 4: The heat map of the attention weights $\alpha_{t',t}$ in grayscale (0: black, 1: white)

5 User Study

This section conducts a user study to analyze the usability of DeepMnemonic, and validates that the mnemonic sen-

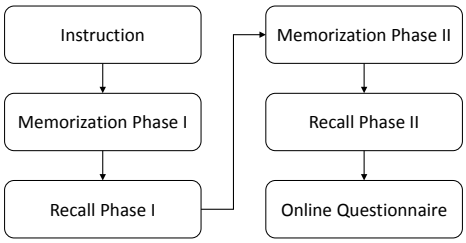


Fig. 5: The user study procedure.

tences generated by DeepMnemonic are helpful for users to memorize passwords.

5.1 User Study Setting

In this user study, we recruited 24 participants from universities and institutes, and randomly divided them into three groups, labeled as *group 0*, *group 1*, and *group 2*. The lengths of passwords assigned to the three groups were 8, 12, and 16, respectively. This user study included three main tasks, i.e., password memorization, password recall, and online questionnaire. The password memorization and password recall tasks were conducted face-to-face in our lab, where we gave task instructions and measured the participants' performance using a timer.

Figure 5 illustrates the user study procedure. In Memorization Phase I without the aid of mnemonic sentences, each participant was asked to memorize an assigned password p_1, where the time the participant used for memorization is t_1. Then, after a period of time, in Recall Phase I, each participant was asked to recall and write down the assigned password p_1, and the recalled password is r_1. In Memorization Phase II with the aid of the mnemonic sentences generated by DeepMnemonic, each participant was asked to memorize a different assigned password p_2, where the time the participant used for memorization is t_2. Later in the Recall Phase II, the recalled version of p_2 is termed as r_2. Finally, the online questionnaire was designed to evaluate the participants' experiences about the whole user study process. Following the forgetting curve theory [38], we set the time gap between password memorization and password recall to 48 hours[1].

5.2 User Study Results

Figure 6 shows the average results over all participants within each group (i.e., with the same password length) with and without the aid of the generated mnemonic sentences by DeepMnemonic.

Figure 6a shows the time costs in memorizing the passwords without (t_1) and with (t_2) the aid of mnemonic sentences. Note that t_2 includes the time for memorizing the associated mnemonic sentence in addition to the password. It is observed that t_2 is lower than t_1, especially for memorizing longer passwords. We conducted a statistical t-test which can work well with our small set of samples that approximate the normal distribution [39] [40]. The difference between the two sets of time costs is statistically significant at a significance level $\alpha = 0.01$. This indicates that, by using the mnemonic sentences generated by DeepMnemonic, the participants are able to memorize the passwords more quickly than without the aid of any mnemonics. For *group 2*, who memorized passwords of length 16, the observed difference is more significant, i.e., 110 seconds versus 42.6 seconds with $p_{\text{value}} = 0.007$. Clearly, DeepMnemonic shows its capability of assisting in memorizing passwords more effectively, and is especially helpful for memorizing relatively long passwords.

Figure 6b evaluates the password recall in terms of *edit distance* between each pair of recalled password r and assigned password p (the smaller, the better). Overall the average edit distance with the aid of DeepMnemonic is smaller than that without using DeepMnemonic. The differences become even larger when the password lengths become longer. For *group 2* where the password length is 16, the average edit distance with the aid of DeepMnemonic is significantly smaller than that without the aid. Our statistical test shows that the difference is significant given the level $\alpha = 0.01$. However, for shorter passwords in *group 0* where password length is 8, the average edit distances with and without using DeepMnemonic are almost the same (0.75). A possible explanation is that, when a given password is short, the efforts used to remember the generated mnemonic sentence and the password itself are comparable. In such case, using mnemonic sentences may incur extra burdens for memorizing short passwords.

This can also be discovered from the complete recall ratio in Figure 6c (the higher, the better). It shows that users can recall 8-character passwords better without the burden of memorizing extra mnemonic sentences. However, the helpfulness of mnemonic sentences becomes obvious for users to memorize 12-character and 16-character passwords. Overall the mnemonic sentences generated by DeepMnemonic are effective in improving the performance of password recall, especially for longer/stronger passwords.

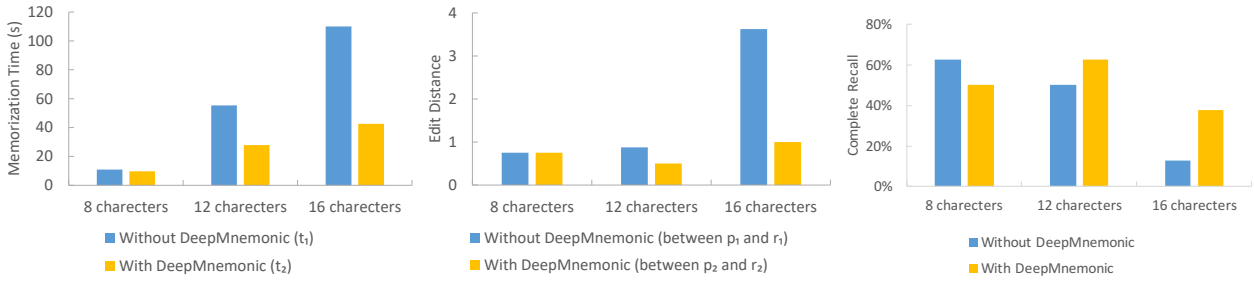
In addition, we analyzed the results of online questionnaire to evaluate the user experience. Figure 7 shows that only 8.3% of the participants indicated that they are *not* perplexed by memorization of passwords, which clearly suggests that remembering passwords is indeed a difficult task. Almost 96% of participants agreed that the passwords assigned to them are secure and strong. About 46% of them considered their assigned passwords easy to remember. We note that the majority of them come from *group 0*, where their assigned passwords are not very long, and are thus not very difficult to remember.

A large portion of the participants agreed that both gist and exact words of each generated mnemonic sentence are helpful and easy to remember. Almost 71% and 67% of participants agreed on the two points, i.e., "gist is easy to remember" and "wording is easy to remember", respectively. With regard to grammatical correctness, about 8.3% of participants noticed a few grammar errors existed in the generated sentences. For example, in Case 6 of Table 3, the simple present tense in the sentence may not be rigorously reasonable. The majority of the participants (70.9%) agreed that the mnemonic sentences generated by DeepMnemonic are meaningful. Overall, 79.2% of participants recognized that DeepMnemonic is generally helpful for remembering the given passwords.

6 Discussion

In this section, we discuss several issues related to the usability of DeepMnemonic.

1. The detailed user study design as well as the ethical consideration can be found at https://goo.gl/KtwuGB



(a) Memorization time cost (b) Edit distance (c) Complete recall

Fig. 6: The comparison results of different user groups with and without using DeepMnemonic. We compare the memorization time cost (a), edit distance (b), and complete recall ratio (c) of the three groups without and with the aid of DeepMnemonic. For each comparison, the results for different password lengths are shown separately.

Strongly disagree	Disagree	Neutral	Agree	Strongly agree
You are perplexed by remembering passwords.				
0.0%	8.3%	37.5%	41.7%	12.5%
The passwords assigned to you during the user study are strong passwords.				
0.0%	4.2%	0.0%	75.0%	20.8%
The passwords assigned to you during the user study are easy to remember.				
4.2%	37.5%	12.5%	33.3%	12.5%
The gist of the mnemonic sentence generated by DeepMnemonic is easy to remember exactly.				
0.0%	0.0%	29.2%	58.3%	12.5%
The wording of the mnemonic sentence generated by DeepMnemonic is easy to remember exactly.				
0.0%	8.3%	25.0%	58.3%	8.3%
The mnemonics generated by DeepMnemonic is grammatically correct.				
0.0%	8.3%	20.8%	45.8%	25.0%
The mnemonics generated by DeepMnemonic is semantically meaningful.				
0.0%	4.2%	25.0%	54.2%	16.7%
DeepMnemonic is generally helpful in remembering the passwords in general.				
0.0%	0.0%	20.8%	62.5%	16.7%

Fig. 7: Online questionnaire analysis results.

6.1 Learning from Multiple Strong Password Generation Rules

In general, the successful application of DeepMnemonic to automatic mnemonic generation largely depends on the large-scale training data used for building the sequence-to-sequence learning model. Recall that in our experiments, we generated pairwise password and mnemonic sentence training data from Webis-Simple-Sentences-17. For each sentence in the dataset, we followed one strong password generation rule proposed in [22], and thus concatenated all the first letter of each word and special characters in the sentence so as to create a password. It is worth noting that DeepMnemonic is not limited to this single strong password generation rule adopted for constructing the pairwise training data. It is possible that DeepMnemonic can also be trained on other datasets constructed using different password generation rules, such as concatenating the last letter of each word instead of the first letters in deriving passwords. Thanks to the use of the *sequence-to-sequence* learning model, DeepMnemonic is able to learn language generation patterns automatically from the training data, and map passwords to mnemonic sentences in testing. Therefore, it is possible to train multiple encoder-decoder language generation models in DeepMnemonic using different password generation rules. As a result, users have flexibility to choose from multiple generated mnemonic sentences for a given password, which may further improve the usability of DeepMnemonic.

6.2 Baseline Selection

The n-gram language model used in our comparison evaluation is one of the most well-known language generation models. We have tested various values of n ($n \in [1, 4]$) in the experiments. Since the bigram language model achieved the best performance among different n, it was selected as the baseline.

We discovered that when using a unigram language model to map from a password sequence to a mnemonic sequence, the model cannot leverage on the context, i.e., preceding words. As a result, the words that appear more frequently in training data are considered more important, and are then assigned with a higher probability for prediction. For example, given a password character "t", it is very likely that the unigram language model would generate the word "the", regardless of the preceding contextual words. In addition, trigram ($n = 3$) and 4-gram based language models built on the pairwise training data often suffer from sparse co-occurrence information, and output the unknown token <UNK>. It is more likely that these models, compared to the bigram models, generate empty list of candidate words from two or more contextual preceding words and a target password character. According to our experimental results, when n is larger than 2, the language models have more difficulty in generating complete sentences.

6.3 Evaluation Metric

BLEU is one of the most popular automated metrics for evaluating the *quality* of machine translation [37] in natural language processing. The quality is often considered to be the matching degree between each pair of generated sentence and ground-truth sentence. The closer the generated sentence is to the ground-truth, the better the language generation is. Our quantitative experimental results show that DeepMnemonic significantly outperforms the well-established bigram language model in terms of BLEU.

We note that BLEU is perhaps a good metric to evaluate the quality of machine translation, where each generated or translated sentence should be close to the reference translation (ground truth) as much as possible. The state-of-the-art BLEU score is 34.8 for machine translation using deep neural networks [35]. However, BLEU may not be a *fair* metric for evaluating the password mnemonic generation, which aims to assist users in password memorization. It is true that the generated mnemonic sentences sometimes do not match perfectly to the reference sentences (ground

truth), which may lead to poor BLEU scores. However, it is possible that such mnemonic sentences are still helpful for memorizing passwords, as long as they are grammatically correct and semantically meaningful by themselves, as shown in Table 3. Another complementary metric MP is thus designed to evaluate the quality of password mnemonic generation.

6.4 User Study Scale and Population

In our user study, we recruited 24 participants to evaluate the usability of DeepMnemonic. Despite the small scale, the user study shows that DeepMnemonic is helpful in assisting users in better memorizing the passwords, especially when passwords are relatively long. A limitation of this user study lies in the age distribution of the participants. The ages of the participants range from 24 to 40, which are not representative of younger or older population. Indeed, this is also a challenge in many other user studies [41]. Our user study suggests that DeepMnemonic is generally helpful for young and middle-aged people. It remains a future work to evaluate the DeepMnemonic against other age groups.

6.5 Lowercase and Uppercase Password Characters

DeepMnemonic does not handle lowercase and uppercase characters when generating mnemonic sentences, and thus cannot help users distinguish between them. We have tried to build a variant DeepMnemonic model, which can generate sentences that differentiate lowercase and uppercase characters. However, its performance (e.g., BLEU and MP) is not as good as that of current DeepMnemonic model. We plan to deal with this lowercase and uppercase character issue in the future work. In addition, users may have multiple passwords across various platforms. It would be interesting to extend DeepMnemonic so that it can help users memorize their passwords across different platforms.

6.6 Generating Mnemonics in Different Languages

Currently, the proposed DeepMnemonic has been shown to be effective for generating mnemonics for the given strong passwords in English. Similar to the *sequence-to-sequence* machine translation task in natural language processing [30], the underlying encoder-decoder model of DeepMnemonic can encode any input password into a semantic vectorial representation and decode the representation into a semantically meaningful mnemonic sentence in the target language. In the case of suitable training datasets being available in other languages (e.g., Chinese), DeepMnemonic can be adapted to the mnemonic sentence generation in a different language.

7 Related Work

7.1 Strong Password Evaluation and Generation

A lot of efforts have been made in the past to assess the strength of passwords or to generate strong passwords. The strength of a password can be typically evaluated by two common metrics, namely, entropy and guessability. The entropy measures how unpredictable a password is by considering the length of password and the distribution of characters in the password. One limitation of the entropy-based measurement [42] is that it only supplies users with rough approximations of password strength [3] [43] [44]. In contrast, guessability-based measurement, which is defined as the number of guesses required to break a password, has become increasingly popular. To evaluate the guessability of a password, one key step is to identify an algorithm for password cracking, such as the Probabilistic Context-Free Grammar model (PCFG) [10] [11] [6] and the Markov n-grams model [12] [4]. These cracking algorithms exploit the password distributions that are derived from various password datasets disclosed in previous security incidents. It is revealed that different password datasets collected from different user groups demonstrate different distributions [45] [46]. Then, the guessability of the password can be measured using the cracking algorithm.

Previous studies have evaluated the password strength by measuring the popularity of passwords. Schechter et al. [47] evaluated user-chosen passwords by identifying undesirably popular passwords. The passwords within certain popularity threshold are considered secure under probabilistic attacks. It is argued that the existing password creation policies can be replaced with popularity limits so as to strengthen both security and usability for user authentication systems.

Strong password generation aims to strengthen passwords via changing or adding characters to the passwords. Generation of persuasive text passwords is one of such approaches, which inserts one to four characters at random positions in a given password [7] [8]. Recently, Houshmand and Aggarwal [9] presented an *analyze-modify* method to generate strong passwords. They first evaluated password strength using PCFG. For identified weak passwords, they modified them based on a set of editing rules. One limitation of strong password generation is that it does not take into consideration the usability or memorability of passwords. As a consequence, the generated passwords may not be user-friendly in practice.

7.2 Memorable Password Generation

A variety of strategies have been employed to create easy-to-remember passwords by service providers. One strategy is to generate pronounceable passwords [16] [18]. For example, pronounceable password "kilakefe52" is comprised of a random sequence of vowel-consonant pairs [16]. Another strategy is to create memorable passwords that comprise multiple components with certain meanings [17]. For example, password "#FreDDi17%" can be divided into three meaningful parts, i.e., "#" (identity), "FreDDi" (name), and "17%" (percentage). The third strategy aims to generate passwords by simply concatenating a list of random words via hyphen character [19]. The benefit of this strategy is twofold: (i) It is usually difficult to guess unrelated words of the generated passwords; and (ii) Longer passwords often lead to higher entropy and security [48]. Although the generated passwords via the aforementioned strategies are memorable, their strengths have not been systematically evaluated yet, and some of them are vulnerable to password attacks [20].

One typical approach to generate memorable yet strong passwords is to rely on meaningful language expressions, making use of specific parts of given reference sentences or phrases according to certain generation rules [21] [20] [22]. For example, password "*Ilwm.*" can be generated by joining the first character of each word and the punctuation in the following sentence: "*I love watching movies.*" The

expression-based passwords generated by this approach are often supposed to be stronger than those selected intuitively by users [13]; given reference sentences or phrases, the usability cost for memorizing the generated passwords is almost comparable to memorizing those intuitive ones. Yang et al. [20] demonstrated that the security level of an expression-based password was largely affected by its generation rules, which was also validated by Kiesel et al. [22]. However, it is not clear how various generation rules at different security levels affect the usability costs for memorizing the generated passwords. Due to users' behavioral tendency toward picking up easy-to-remember reference sentences [7], the generated memorable passwords using such human-selected language expressions may be easy-to-guess by attackers.

Researchers have studied various tips for creating mnemonic passwords, including sentence substitution (SenSub), key-board change (KbCg), using a formula (UsForm), and special character insertion (SpIns) [49]. Alphapwd [50] is a memorable password generation strategy based on password shapes. In order to create memorable passwords, Alphapwd requires user to remember a shorter sequence of letters that are shown in larger size on top of a normal keyboard; a user may derive his/her strong password easily from the keyboard following the strokes of the shorter sequence of large-size letters. The strengths of these generated passwords are evaluated using probabilistic attack algorithms such as the PCFG algorithm and MarKov model trained on previous disclosed password datasets. Ghazvininejad and Knight [51] proposed to generate passwords in the form of English sequences, called *passphrase*, whose lengths range from 31.2 to 87.7 characters on average. However, experiments revealed that it is difficult for common users to reproduce the exact wording of passphrases, even if they manage to remember the gist of these sentences [52]. Being complementary to these memorable password generation methods, DeepMnemonic is designed to generate mnemonic sentences for any given passwords instead of generating new passwords.

7.3 Password Mnemonics

To improve the memorability of given strong passwords, various types of hints or mnemonic tools have been used. Atallah et al. [53] is the first to propose the concept of using funny jingles for memorizing a randomly generated password. One challenge of this proposal is the generation of related funny jingles. Several other efforts were made to assist users in remembering passwords via external tools, such as helper card [23], hint image [24] [25], and engaging game [26]. Jeyaraman and Topkara [27] proposed to match given passwords to textual headlines or their variant versions selected from a given corpus to assist users in remembering passwords. One drawback of this approach is that the variant headlines generated may be syntactically incorrect or semantically inconsistent. Due to the limited length of the headline text, this approach can only generate hints for short passwords and largely fails for long and strong passwords. The memorability of the passwords generated by this approach has not been evaluated.

7.4 Natural Language Generation

In natural language processing, statistical language models are to generate meaningful sentences by computing joint probabilities of sequences of words from a dictionary of a given language. One of the dominant methods for probabilistic language modeling is the n-gram language model [36], which is a non-parametric learning algorithm. It relies on the preceding sequence of $n-1$ words to estimate the conditional probability for predicting (generating) the current n-th word.

Recently, neural network based language models have become increasingly popular in natural language generation tasks. Bengio et al. [54] proposed a generic neural probabilistic language model, which can simultaneously learn the distributed representation of each word and joint probability function of sequences. Sutskever et al. [35] proposed a sequence-to-sequence neural language model to address the machine translation problem. One key benefit of their model is that it can automatically generate a translated sentence in the target language, given a sentence in the source language. In order to improve machine translation, Bahdanau et al. [30] introduced an attentive alignment strategy to enhance the sequence-to-sequence neural language model, which learned to dynamically pay more attention to salient parts of the input sentences when generating the translated sentences.

In this paper, we exploit natural language translation techniques to generate human-readable and semantically meaningful mnemonic sentences from any given passwords so as to help users memorize strong passwords.

8 Conclusion

In this work, we have proposed DeepMnemonic, a deep neural network based approach to automatic generation of mnemonic sentences for any given textual passwords. DeepMnemonic builds upon an attentive encoder-decoder language generation framework, and works by *translating* an input sequence of password characters to a natural language sentence of mnemonic words. DeepMnemonic is designed to bridge the gap between the strong password generation and the usability of strong passwords. Experimental results show that DeepMnemonic is capable of generating semantically meaningful mnemonic sentences. A user study is conducted to evaluate the usability of DeepMnemonic, which shows that the generated mnemonic sentences are helpful in memorizing strong passwords. Specifically, with the aid of DeepMnemonic, the time used for remembering a password is largely reduced, and the password recall quality is also significantly improved. In the future, we plan to train DeepMnemonic using more comprehensive and diverse training data.

References

[1] R. W. Proctor, M.-C. Lien, K.-P. L. Vu, E. E. Schultz, and G. Salvendy, "Improving computer security for authentication of users: Influence of proactive password restrictions," *Behavior Research Methods*, vol. 34, no. 2, pp. 163–169, 2002.

[2] B. Ur, F. Noma, J. Bees, S. M. Segreti, R. Shay, L. Bauer, N. Christin, and L. F. Cranor, "I added !at the end to make it secure: Observing password creation in the lab," in *Proc. SOUPS*, 2015.

[3] M. Weir, S. Aggarwal, M. Collins, and H. Stern, "Testing metrics for password creation policies by attacking large sets of revealed passwords," in *Proceedings of the 17th ACM conference on Computer and communications security*. ACM, 2010, pp. 162–175.

[4] J. Ma, W. Yang, M. Luo, and N. Li, "A study of probabilistic password models," in *Security and Privacy (SP), 2014 IEEE Symposium on*. IEEE, 2014, pp. 689–704.
[5] W. Melicher, B. Ur, S. M. Segreti, S. Komanduri, L. Bauer, N. Christin, and L. F. Cranor, "Fast, lean, and accurate: Modeling password guessability using neural networks." in *USENIX Security Symposium*, 2016, pp. 175–191.
[6] D. Wang, D. He, H. Cheng, and P. Wang, "fuzzypsm: A new password strength meter using fuzzy probabilistic context-free grammars," in *46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), 2016*. IEEE, 2016, pp. 595–606.
[7] A. Forget, S. Chiasson, P. C. van Oorschot, and R. Biddle, "Improving text passwords through persuasion," in *Proceedings of the 4th symposium on Usable privacy and security*. ACM, 2008, pp. 1–12.
[8] ——, "Persuasion for stronger passwords: Motivation and pilot study," in *International Conference on Persuasive Technology*. Springer, 2008, pp. 140–150.
[9] S. Houshmand and S. Aggarwal, "Building better passwords using probabilistic techniques," in *Proceedings of the 28th Annual Computer Security Applications Conference*. ACM, 2012, pp. 109–118.
[10] M. Weir, S. Aggarwal, B. De Medeiros, and B. Glodek, "Password cracking using probabilistic context-free grammars," in *Security and Privacy, 2009 30th IEEE Symposium on*. IEEE, 2009, pp. 391–405.
[11] R. Veras, C. Collins, and J. Thorpe, "On semantic patterns of passwords and their security impact." in *NDSS*, 2014.
[12] C. Castelluccia, M. Dürmuth, and D. Perito, "Adaptive password-strength meters from markov models." in *NDSS*, 2012.
[13] J. Yan, A. Blackwell, R. Anderson, and A. Grant, "Password memorability and security: Empirical results," *IEEE Security & privacy*, vol. 2, no. 5, pp. 25–31, 2004.
[14] J. R. Anderson and C. J. Lebiere, *The atomic components of thought*. Psychology Press, 2014.
[15] G. A. Miller, "The magical number seven, plus or minus two: Some limits on our capacity for processing information." *Psychological review*, vol. 63, no. 2, p. 81, 1956.
[16] WarpConduit_Computing, "Password generator," https://www.warpconduit.net/password-generator/, 2017.
[17] R. Accettura, "Safepassword," https://www.safepasswd.com/, 2017.
[18] Bad_Neighborhood, "Mnemonic strong password generator," http://www.bad-neighborhood.com/password-generator.htm, 2017.
[19] edoceo, "Mnemonic password generator," http://edoceo.com/utilitas/mnemonic-password-generator, 2017.
[20] W. Yang, N. Li, O. Chowdhury, A. Xiong, and R. W. Proctor, "An empirical study of mnemonic sentence-based password generation strategies," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2016, pp. 1216–1229.
[21] C. Kuo, S. Romanosky, and L. F. Cranor, "Human selection of mnemonic phrase-based passwords," in *Proceedings of the Second Symposium on Usable Privacy and Security*, ser. SOUPS '06. New York, NY, USA: ACM, 2006, pp. 67–78. [Online]. Available: http://doi.acm.org/10.1145/1143120.1143129
[22] J. Kiesel, B. Stein, and S. Lucks, "A large-scale analysis of the mnemonic password advice," in *Proceeding of NDSS*, 2017.
[23] U. Topkara, M. J. Atallah, and M. Topkara, "Passwords decay, words endure: Secure and re-usable multiple password mnemonics," in *Proceedings of the 2007 ACM symposium on Applied computing*. ACM, 2007, pp. 292–299.
[24] M. Fukumitsu, T. Katoh, B. B. Bista, and T. Takata, "A proposal of an associating image-based password creating method and a development of a password creating support system," in *Advanced Information Networking and Applications (AINA), 2010 24th IEEE International Conference on*. IEEE, 2010, pp. 438–445.
[25] K. A. Juang, S. Ranganayakulu, and J. S. Greenstein, "Using system-generated mnemonics to improve the usability and security of password authentication," in *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, vol. 56, no. 1. SAGE Publications Sage CA: Los Angeles, CA, 2012, pp. 506–510.
[26] J. Doolani *et al.*, "Improving memorization and long term recall of system assigned passwords," Ph.D. dissertation, 2016.
[27] S. Jeyaraman and U. Topkara, "Have the cake and eat it too-infusing usability into text-password based authentication systems," in *21st Annual Computer Security Applications Conference*. IEEE, 2005, pp. 10–pp.
[28] P. University, "Wordnet," https://wordnet.princeton.edu/.
[29] G. H. Bower, "Analysis of a mnemonic device: Modern psychology uncovers the powerful components of an ancient system for improving memory," *American Scientist*, vol. 58, no. 5, pp. 496–510, 1970.
[30] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," in *Proceedings of the 2015 International Conference on Learning Representations (ICLR)*, 2015.
[31] M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks," *IEEE Transactions on Signal Processing*, vol. 45, no. 11, pp. 2673–2681, 1997.
[32] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1724–1734.
[33] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.
[34] P. Koehn, "Pharaoh: a beam search decoder for phrase-based statistical machine translation models," in *Conference of the Association for Machine Translation in the Americas*. Springer, 2004, pp. 115–124.
[35] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Advances in neural information processing systems*, 2014, pp. 3104–3112.
[36] C. D. Manning, C. D. Manning, and H. Schütze, *Foundations of statistical natural language processing*. MIT press, 1999.
[37] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "Bleu: a method for automatic evaluation of machine translation," in *Proceedings of the 40th annual meeting on association for computational linguistics*. Association for Computational Linguistics, 2002, pp. 311–318.
[38] H. Ebbinghaus, "Memory: A contribution to experimental psychology," *Annals of neurosciences*, vol. 20, no. 4, p. 155, 2013.
[39] T. G. Dietterich, "Approximate statistical tests for comparing supervised classification learning algorithms," *Neural computation*, vol. 10, no. 7, pp. 1895–1923, 1998.
[40] N. K. Bakirov and G. J. Szekely, "Students t-test for gaussian scale mixtures," *Journal of Mathematical Sciences*, vol. 139, no. 3, pp. 6497–6505, 2006.
[41] Y. Li, Y. Cheng, Y. Li, and R. H. Deng, "What you see is not what you get: Leakage-resilient password entry schemes for smart glasses," in *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*. ACM, 2017, pp. 327–333.
[42] NIST, "Digital identity guidelines," https://pages.nist.gov/800-63-3/, 2017.
[43] P. G. Kelley, S. Komanduri, M. L. Mazurek, R. Shay, T. Vidas, L. Bauer, N. Christin, L. F. Cranor, and J. Lopez, "Guess again (and again and again): Measuring password strength by simulating password-cracking algorithms," in *Security and Privacy (SP), 2012 IEEE Symposium on*. IEEE, 2012, pp. 523–537.
[44] B. Ur, P. G. Kelley, S. Komanduri, J. Lee, M. Maass, M. L. Mazurek, T. Passaro, R. Shay, T. Vidas, L. Bauer *et al.*, "How does your password measure up? the effect of strength meters on password creation." in *USENIX Security Symposium*, 2012, pp. 65–80.
[45] D. Wang, P. Wang, D. He, and Y. Tian, "Birthday, name and bifacial-security: understanding passwords of chinese web users," in *28th {USENIX} Security Symposium ({USENIX} Security 19)*, 2019, pp. 1537–1555.
[46] D. Wang, Q. Gu, X. Huang, and P. Wang, "Understanding human-chosen pins: characteristics, distribution and security," in *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*. ACM, 2017, pp. 372–385.
[47] S. Schechter, C. Herley, and M. Mitzenmacher, "Popularity is everything: A new approach to protecting passwords from statistical-guessing attacks," in *Proceedings of the 5th USENIX conference on Hot topics in security*. USENIX Association, 2010, pp. 1–8.
[48] R. Shay, P. G. Kelley, S. Komanduri, M. L. Mazurek, B. Ur, T. Vidas, L. Bauer, N. Christin, and L. F. Cranor, "Correct horse battery staple: Exploring the usability of system-assigned passphrases," in *Proceedings of the Eighth Symposium on Usable Privacy and Security*, ser. SOUPS '12. New York, NY, USA: ACM, 2012, pp. 7:1–7:20. [Online]. Available: http://doi.acm.org/10.1145/2335356.2335366
[49] B. Ye, Y. Guo, L. Zhang, and X. Guo, "An empirical study of mnemonic password creation tips," *Computers & Security*, vol. 85, pp. 41–50, 2019.
[50] J. Song, D. Wang, Z. Yun, and X. Han, "Alphapwd: A password generation strategy based on mnemonic shape," *IEEE Access*, vol. 7, pp. 119 052–119 059, 2019.

[51] M. Ghazvininejad and K. Knight, "How to memorize a random 60-bit string," *Parking*, vol. 11, no. 70.8, pp. 58–83, 2015.
[52] H. A. Yajam, Y. K. Ahmadabadi, and M. Akhaee, "Papiapass: Sentence-based passwords using dependency trees," in *13th International Iranian Society of Cryptology Conference on Information Security and Cryptology (ISCISC)*. IEEE, 2016, pp. 91–96.
[53] M. J. Atallah, C. J. McDonough, V. Raskin, and S. Nirenburg, "Natural language processing for information assurance and security: an overview and implementations," in *Proceedings of the 2000 workshop on New security paradigms*. ACM, 2001, pp. 51–65.
[54] Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin, "A neural probabilistic language model," *Journal of machine learning research*, vol. 3, no. Feb, pp. 1137–1155, 2003.

Yao Cheng is currently a senior researcher at Huawei International in Singapore. She received her Ph.D. degree from University of Chinese Academy of Sciences. Her research interests include security and privacy in deep learning systems, blockchain technology applications, Android framework vulnerability analysis, mobile application security analysis, and mobile malware detection.

Chang Xu is currently a Postdoctoral Fellow at Data61, CSIRO, Australia. He received his PhD degree in Computer Science from Nanyang Technological University in March 2017. His current interests include robust and explainable deep neural models for natural language processing.

Zhen Hai received the PhD degree in Computer Science and Engineering from Nanyang Technological University, Singapore. He is a Research Scientist in 2014 at the Institute for Infocomm Research, A*STAR, Singapore. His research interests include natural language processing, text mining, sentiment analysis, information security, and machine learning. He has been invited to serve on the program committees of leading conferences including SIGIR, ACL, AAAI, IJCAI, CIKM, etc.

Yingjiu Li is currently a Ripple Professor in the Computer and Information Science Department at the University of Oregon. His research interests include IoT Security and Privacy, Mobile and System Security, Applied Cryptography and Cloud Security, and Data Application Security and Privacy. He has published over 140 technical papers in international conferences and journals, and served in the program committees for over 80 international conferences and workshops, including top-tier cybersecurity conferences and journals.