Using Neural Network Chains to Launch a Preimage Attack on Reduced-Round SHA-1

Grant Proposal

Erica Dong

Massachusetts Academy of Math and Science

Worcester, MA

Abstract

As the world becomes increasingly online, with the development of technologies such as the Internet of Things, digital protection becomes more and more important. Cryptographic hashes are a class of algorithms that are critical to this goal. These algorithms have certain ways of being analyzed and exploited with techniques from the field of cryptanalysis, which deepens understanding and identifies how to strengthen them. One understudied area of cryptanalysis is neuro-cryptanalysis, or cryptanalysis using neural networks. In particular, its potential in launching preimage attacks has been overlooked. Previous research using this approach has introduced too much complexity onto a single network, lowering efficiency and consequently weakening the efficacy of the attacks. Instead, by training several neural networks, one for each round of the hash, each neural network needs to model less complexity, significantly improving efficiency. These networks can be linked together to create a full attack on the hash. As a proof of concept, the SHA-1 hash was selected because it has a compatible architecture and is well-studied. An attack on reduced-round, one-block SHA-1 will be conducted first before applying extensions and optimizations.

Keywords: Cryptanalysis, neuro-cryptanalysis, neural networks, hash functions, SHA-1, preimage attacks

Using Neural Network Chains to Reverse the Avalanche Effect in Reduced-Round SHA-1

With the advent of the information age, there is a rising need for digital security. Personal information, medical data, passwords, and IoT devices are just a few examples of critical information systems that need to be protected. Cryptographic hashes are essential to this goal. These deterministic functions map a variable-length message (the plaintext) to a fixed-length hash value (the ciphertext) (Fig. 1), but in a manner that is difficult to reverse and appears random. These are used to securely encrypt information. For example, when a user enters a new password to a website, that password is first hashed before being saved in the database. When that user wants to log in again, the entered password is also hashed and compared to the saved hash. Then, even if a hacker breaks into the database, the original password is not discoverable, so the user's account is safe. Hence, it is critical that cryptographic hashes continue to be improved upon as technologies advance to ensure a safer Internet.

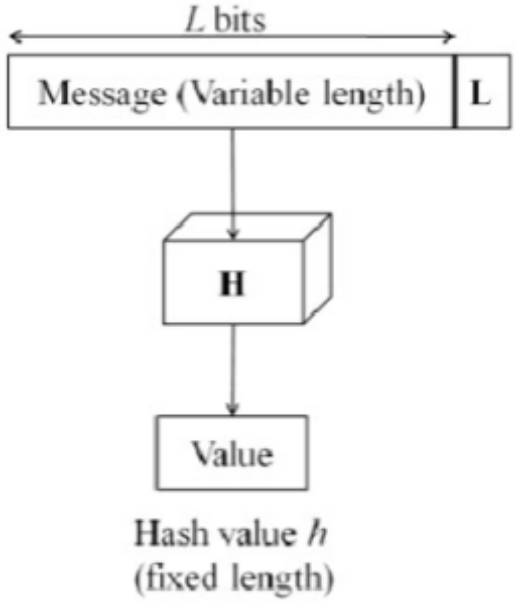

Fig. 1. Structure of a hash function (Sharma & Bhatt, 2018)

Past Work

Hashes can only be improved by first finding their weak points. As an analogy, a leaky pipe can only be fixed once the hole is identified. This area of study, which involves analyzing and exploiting cryptographic systems, is called cryptanalysis. One subsection of cryptanalysis is neuro-cryptanalysis, or cryptanalysis using machine learning. Although this field is relatively small, there have been some advances made over the past decade. First, in his seminal 1993 paper, Rivest described the potential links between machine learning and cryptanalysis, going so far as to dub them "sister fields." In 2012, Alani conducted a known-plaintext attack using a cascading neural network trained on

plaintext-ciphertext pairs to predict the plaintext of the DES and Triple-DES ciphers based on the ciphertext without knowing the secret key. This attack successfully reduced both the time and data needed to conduct a known-plaintext attack on these ciphers. In 2017, Greydanus used recurrent neural networks to successfully, albeit inefficiently, reverse polyalphabetic ciphers, specifically the Vignere, Autokey, and Enigma ciphers, which are much weaker than modern cryptographic hashes. In 2019, Goncharov trained a standard feed-forward neural network on plaintext-ciphertext pairs from several common cryptographic hashes while making use of fuzzy logic, which extends boolean values to all values between 0 and 1, but only achieved success in extremely low-round versions of the hashes. Also in 2019, Gohr presented a new strategy using deep neural networks that distinguished between random data and outputs produced by a reduced-round Speck cipher given an input difference, greatly aiding in differential cryptanalysis. This neural distinguisher surpassed state-of-the-art techniques in the amount of rounds it successfully attacked. In 2020, So successfully launched a known-plaintext attack on SDES, Simon, and Speck by restricting the keyspace to 64 ASCII characters, skewing the key bit distribution. In 2021, Benamira et al. built upon this work by translating the deep neural network into pure cryptanalysis, creating an optimized version of the machine learning model, and further improving it through modifications such as adding batches. In the same year, Liu et al. used deep neural networks to launch a preimage attack against the reduced-round Xoodyak hash by training them on plaintext-ciphertext pairs, achieving limited success in one-round Xoodyak. Overall, neuro-cryptanalytic techniques have achieved widely varying levels of success in the past. However, most of this research used neural networks without considering the hash's internal structure, which may make their approaches less effective.

Research

Most past research has failed to consider the algorithms' architecture, instead using neural networks as black-boxes to try to cover the entire function at once. By breaking the algorithm into its

components and linking together several neural networks to analyze each, greater efficiency and accuracy could potentially be achieved. Therefore, the goal of this project is to construct a system to apply neural network chains to conduct a preimage attack on reduced-round SHA-1. A preimage attack is the quintessential cryptanalytic attack, where the attacker finds the plaintext based on the ciphertext of a hash. The chosen attack model is Secure Hashing Algorithm 1, or SHA-1, due to its suitable architecture and the extensive research and documentation on it. A successful attack would serve as a proof-of-concept for a new approach to cryptanalysis.

Section II: Specific Aims

This project's objective is to construct a system that applies neural network chains to conduct a preimage attack on reduced-round one-block SHA-1. The long-term goal is to identify weaknesses in cryptographic hashes, particularly ones that can be exploited by neural networks, so that these algorithms can be strengthened. Strengthening cryptographic hashes subsequently contributes to a safer future in an era where people are increasingly reliant on digital technology. The central hypothesis of this proposal is that linking together multilayer feed-forward neural networks that are individually trained on data from each round of SHA-1 will result in an attack on the full hash. The rationale is that, since this type of neural network is a universal approximator, it will be able to model the function that reverses a specific round of the hash (Hornik et al., 1989). However, this method would be more effective than trying to reverse the whole hash with one network, since each neural network has to handle much less complexity. SHA-1 was chosen because it is well-studied, its architecture enables easy generation of the necessary data, and its round operations are easily interpretable. However, the approach is applicable to any Merkle–Damgård hash.

Specific Aims

The work proposed aims to accomplish three specific goals. First, the neural networks for each individual round should achieve adequate accuracy, preferably above 90%. Second, these networks will be linked together, and plaintext should be extracted correctly, with nontrivial accuracy. Third, the networks will be optimized and the chain length extended until trivial accuracy is reached to scope out the full potential of the method. The expected outcome of this work is a successful preimage attack on reduced-round SHA-1 based on a novel neuro-cryptanalytic approach, which will open doors for stronger attacks in the future.

Section III: Project Goals and Methodology

Relevance

The world is increasingly online. From smart cars to digital transactions to remote work and education, there are more and more services that rely on cryptographic algorithms to keep them safe, with dire consequences if compromised. Hence, as attackers gain access to more advanced technology and greater computing power, it is critical that these algorithms are strengthened. Although counterintuitive, the best way to do this is to first identify their weaknesses. This project develops a novel approach to attacking cryptographic hashes using neural network chains.

Innovation

Most previous work modeled the entire hash with a single neural network, failing to consider the internal structure of the algorithm. This resulted in extremely low preimage accuracy. One such study on the hash Xoodyak only achieved genuine success in one-round modified versions of the algorithm, which is incredibly weak (Liu et al., 2021). Another experiment, conducted on SHA-1, MD-5, SHA-2, and SHA-3, similarly only found relative success with versions weakened to one round (Goncharov, 2019). Both

researchers used one neural network per hash and trained them directly on randomly-generated plaintext and the corresponding ciphertext.

This project will improve the strength of this approach past a single round by using neural network chains and giving them more information to utilize in the form of intermediate hash states, which will provide a proof-of-concept for an underestimated and overlooked cryptanalytic method.

Methodology

Data Generation

First, data will be generated on the SHA-1 hash using a custom implementation in Java. Binary input data will be randomly generated. The length of these inputs will be 432 to ensure that the data is processed in one 512-bit block while simultaneously accounting for padding (Dang, 2015). This project will only deal with one-block inputs, since each additional block effectively adds 80 more rounds to the hash. As the method's ability to deal with more than one round is unknown, dealing with multiple blocks would be outside the scope of the experiment. The inputs will be produced using Java's Random class, which provides pseudorandom numbers. To save time, they will not be checked for duplicates, despite this random generation. Since there are 2^{512} possible binary numbers of length 512 or less, collisions are extremely unlikely.

Then, this input data is fed through the custom SHA-1 implementation. Instead of only the output, this algorithm will return an entire history of the hash's internal state as it evaluates the input data from round 1 to round 80, which will be saved in CSV format along with the original input. 80% of this data will be used for training, while the other 20% will be used for testing.

Neural Network Creation and Training

A multilayer feed-forward neural network template will be created in Python using the PyTorch AI library. The starting model will most likely have 2 hidden layers, which is relatively shallow. The input and output layers will be size 160, since SHA-1's internal state is always 160 bits (Dang, 2015). Other architectural aspects and hyperparameters will be optimized by referencing best practices, utilizing results from past papers, and adjusting based on training results. In this regard, past papers are extremely useful. Goncharov (2019) noted that batch normalization greatly improved accuracy, halving the average number of incorrect bits, while occasional regularization techniques and binary cross-entropy loss yielded slight, unspecified improvements in accuracy. He also focused on fuzzy logic as a possible aid to gradient descent since it provides the needed continuity. Benamira et al. (2021) also discussed batches as a major performance-improver in neuro-cryptanalysis, although for a neural distinguisher rather than an approximator. Liu et al. (2021) used the ReLu activation function for their hidden layers and the sigmoid function for the output layer, after finding that sigmoid performed better than softmax. These recommendations will be combined to create the initial neural network. Parameters will then be adjusted based on performance.

Once the neural network template is created, it will be trained on different sets of round data. Let NN_x be the neural network that transforms from round x to round x-1. For example, NN_3 reverses round 3 to round 2. Then, each NN_x is trained to map inputs from round x to their corresponding outputs from round x-1. As mentioned before, hyperparameters and architecture will be altered based on accuracy and performance.

Linking and Postprocessing

After the neural networks have reached adequate accuracy, they will be linked together so that the output of one is the input of another. For example, NN_3 will feed into NN_2. Since each round is transformed by a part of the plaintext in the forward direction, different parts of the plaintext can be

reconstructed from this round history algorithmically. If the chain is at least 16, the entire plaintext can be reconstructed due to SHA-1's architecture: the plaintext is split into 16 words of bit length 32, which are used in the first 16 rounds (Dang, 2015). The 64 other rounds use data extended from those 16 words. Short chains will be tested first, and more rounds will be added based on success rate, where success is defined as producing a correct part of the plaintext.

Enhancement Using Other Techniques

The attack could possibly be augmented by applying techniques from meet-in-the-middle (MITM) attacks, which are effective against SHA-1. In particular, one paper presented at the conference CRYPTO 2012 combined MITM attacks with differential cryptanalysis to greatly strengthen reduced-round SHA-1 preimage attack performance (Knellwolf & Khovratovich, 2012). MITM is a technique where a secret key is deduced (in SHA-1, the plaintext acts as the key) by dividing the hash function into blocks and, starting from the initial value and the ciphertext, working towards the middle while validating guesses on the secret key. This strategy is similar to the neural network chains, which also divide the hash function and guess information as they work their way through the hash to a known initial value. Application of successful concepts from the Knellwolf and Khovratovich paper may augment the performance of the neural network chain attack significantly.

Overall Justification and Feasibility

Choice of SHA-1

As mentioned before, SHA-1 was chosen due to its compatible architecture. Its internal state stays a constant size, making neural network training easier, and it processes the input data in only one way, whereas other hashes may have multiple stages. It was chosen above its successors, SHA-2 and SHA-3, due to its greater simplicity. In addition, it is suitably strong. One study investigating several

standard hashes, including SHA-1 and its successors, on their randomness and non-correlation, found that SHA-1 held up well, even slightly outperforming SHA-2 and SHA-3 on certain tests (Upadhyay et al., 2022). It is also a former NIST standard. Furthermore, since it is so well-studied, there is an abundance of past work on its robustness against various attacks, providing a good baseline.

Computational Feasibility

Past work from Liu et al. (2021) and Goncharov (2019) found neural networks effective for the single-round versions of the hashes they studied. That is essentially analogous to an individual neural network in this project since it reverses one round of the hash. Training a single neural network should be faster for this project since a single-round permutation is even simpler than a single-round hash. Although it is necessary to train several neural networks, this can be done in parallel through a GPU. Hence, the work proposed is completely computationally feasible.

Specific Aims

Specific Aim #1

The first aim of the project is to train individual neural networks to reverse rounds of SHA-1 with adequate accuracy, preferably above 90%.

Justification and Feasibility. As described in the methodology, data will be generated on each round using a modified version of SHA-1. Then, the neural networks will be built and trained on this data. They will most likely be able to reach a good accuracy since the transformation applied in a single round of the algorithm is much easier to model than the entire algorithm. In fact, Liu et al. (2021)

Fig. 2. The number of correctly predicted messages for each margin of incorrect bits for three Xoodyak attack models (Liu et al., 2021)

successfully trained neural networks for single-round Xoodyak with a decent level of exact accuracy, demonstrating this aim's feasibility (Fig. 2).

Expected Outcome. The overall outcome of this aim is to achieve success on Specific Aim #2, which hinges on the individual neural networks being successful so that they can be linked together.

Potential Pitfalls and Alternative Strategies. If neural network accuracy is low starting out, which is plausible since round transformations are designed to appear random, the many recommendations from past literature can be applied, such as the fuzzy logic discussed by Goncharov (2019).

Specific Aim #2

The second objective of this project is to link together the neural networks constructed for Specific Aim #1, where the plaintext fragments extracted from the chain are correct with nontrivial accuracy.

Justification and Feasibility. If Specific Aim #1 is successful, then it should be feasible to achieve good accuracy for at least short neural network chains, as described in the methodology. Even when only a single neural network is used, as described by Goncharov (2019), two rounds of SHA-1 can be successfully modeled (Fig. 3). Although that model is not the best, it still performs significantly better than random. Since the aim is to link together two neural networks that each successfully model one round, rather than push a single network to model two rounds, the chain model should be even more robust than Goncharov's.

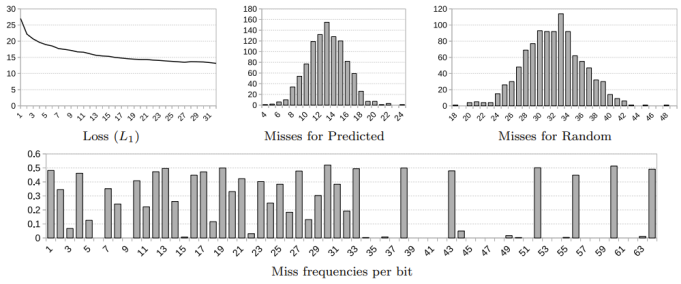

Fig. 3. Data on neural network results for two-round SHA-1 (Goncharov, 2019)

Expected Outcomes. The overall outcome of this aim is to produce a preimage attack on severely reduced-round SHA-1 by feeding the ciphertext into a short neural network chain and subsequently extracting plaintext from it, which sets the stage for Specific Aim #3.

Potential Pitfalls and Alternative Strategies. The chain length is expected to be very short, since percent accuracy will decrease multiplicatively. For example, if one neural network has an accuracy of 90%, and the next also has an accuracy of 90%, then the combined output accuracy will only be (0.9)(0.9) = 0.81 = 81%. To remedy this, further optimizations to base neural network performance can be applied by experimenting with various techniques and applying the recommendations of past literature.

Specific Aim #3

The final objective of this project is to extend neural network chain length until trivial accuracy is achieved and apply possible extensions such as meet-in-the-middle (MITM) attack techniques.

Justification and Feasibility. Extending chain length until trivial accuracy is simply a matter of linking together more neural networks, which should be entirely feasible. Applying MITM attack techniques should also be feasible. As seen in the diagram from Knellwolf and Khovratovich (2012), MITM attacks also rely on breaking up the hash algorithm into components, modeling those components, and transferring data between them to infer the plaintext (Fig. 4). The MITM strategic framework is similar to the principles behind the neural network chain approach, so translating techniques is likely possible.

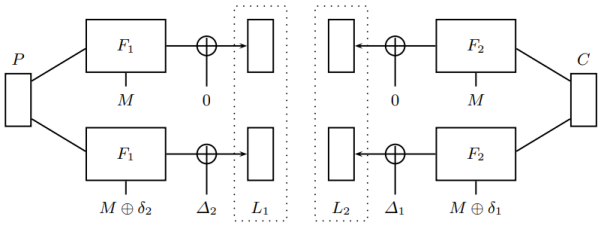

Fig. 4. Model of an MITM attack (Knellwolf & Khovratovich, 2012)

Expected Outcomes. The overall outcome of this aim is to identify the limits of the neural network chaining technique by pushing it until it is no longer effective and to connect the technique with other areas of cryptanalysis.

Potential Pitfalls and Alternative Strategies. One potential pitfall of this aim is that MITM and neural network chaining are not guaranteed to have techniques that can readily be translated. Although this would be unfortunate, it does not detract from the core aim of developing cryptanalytic techniques from neural network chains.

Section III: Resources/Equipment

Since the project is heavily theoretical, not many resources are required. A laptop will likely be used for most of the experimentation. Utilized software will include Java and Python, as well as the VSCode IDE. In addition, computing power, such as a virtual machine or GPU, may be needed for some of the more computationally heavy portions, primarily the neural network training.

Section V: Ethical Considerations

As this project is primarily software-based, there are barely any ethical considerations. Even though the project attacks a cryptographic algorithm, SHA-1 has already been deprecated due to weaknesses against collision attacks, so even if the attack is successful, minimal harm will be done.

Section VI: Timeline

Build - 11/4 - 12/12

1. Generate data
2. Build the neural network template
3. Begin gathering pre-data

4. Build data pipelines
    a. Link neural networks
    b. Extract preimages
5. Train neural networks

Test and Optimize - 12/13 - 1/21

1. Evaluate the neural networks on a variety of parameters such as strict correctness, Hamming distance, etc.
2. Test different hyperparameters such as neural network depth, loss function, SHA-1 round number, etc.
3. Test various optimization approaches such as the ones described by previous literature
4. If time permits, try on a different hash

Extend and Finalize - 1/21 - 2/8

1. Apply techniques from MITM attacks to improve performance
2. Design graphical abstract and other data visualizations
3. Create tri-fold board
4. Write and practice presentation

Section VII: Appendix

Section VIII: References

Alani, M. M. (2012). Neuro-cryptanalysis of DES and triple-DES. In T. Huang, Z. Zeng, C. Li, & C. S. Leung (Eds.), *Neural Information Processing* (Vol. 7667, pp. 637–646). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-34500-5_75

Benamira, A., Gerault, D., Peyrin, T., & Tan, Q. Q. (2021). A deeper look at machine learning-based cryptanalysis. In A. Canteaut & F.-X. Standaert (Eds.), *Advances in Cryptology – EUROCRYPT 2021* (Vol. 12696, pp. 805–835). Springer International Publishing. https://doi.org/10.1007/978-3-030-77870-5_28

Dang, Q. H. (2015). *Secure Hash Standard* (NIST FIPS 180-4; p. NIST FIPS 180-4). National Institute of Standards and Technology. https://doi.org/10.6028/NIST.FIPS.180-4

Gohr, A. (2019). Improving attacks on round-reduced Speck32/64 using deep learning. In A. Boldyreva & D. Micciancio (Eds.), *Advances in Cryptology – CRYPTO 2019* (Vol. 11693, pp. 150–179). Springer International Publishing. https://doi.org/10.1007/978-3-030-26951-7_6

Goncharov, S. V. (2019). *Using fuzzy bits and neural networks to partially invert few rounds of some cryptographic hash functions*. arXiv. https://doi.org/10.48550/ARXIV.1901.02438

Hornik, K., Stinchcombe, M., & White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks*, *2*(5), 359–366. https://doi.org/10.1016/0893-6080(89)90020-8

Knellwolf, S., & Khovratovich, D. (2012). New preimage attacks against reduced SHA-1. In R. Safavi-Naini & R. Canetti (Eds.), *Advances in Cryptology – CRYPTO 2012* (Vol. 7417, pp. 367–383). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-32009-5_22

Liu, G., Lu, J., Li, H., Tang, P., & Qiu, W. (2021). Preimage attacks against lightweight scheme Xoodyak based on deep learning. In K. Arai (Ed.), *Advances in Information and Communication* (Vol. 1364, pp. 637–648). Springer International Publishing. https://doi.org/10.1007/978-3-030-73103-8_45

Sharma, N., & Bhatt, R. (2018). Privacy preservation in WSN for healthcare application. *Procedia*

Computer Science, *132*, 1243–1252. https://doi.org/10.1016/j.procs.2018.05.040

Upadhyay, D., Gaikwad, N., Zaman, M., & Sampalli, S. (2022). Investigating the avalanche effect of various cryptographically secure hash functions and hash-based applications. *IEEE Access*, *10*, 112472–112486. https://doi.org/10.1109/ACCESS.2022.3215778