Project Notes:

Project Title: Using Neural Network Chains to Launch a Preimage Attack on Reduced-Round SHA-1
Name: Erica Dong

Note Well: There are NO SHORT-cuts to reading journal articles and taking notes from them. Comprehension is paramount. You will most likely need to read it several times, so set aside enough time in your schedule.

Contents:

Knowledge Gaps:

This list provides a brief overview of the major knowledge gaps for this project, how they were resolved and where to find the information.

Knowledge Gap	Resolved By	Information is located	Date resolved
Race/nationality bias in AI	Reading various general articles and papers	Article #1 and #5	10/31
How machine learning is used in cryptography and cryptanalysis currently	Reading a survey paper, skimming over different paper headings	Article #4	9/4
Areas of opportunity/further research in machine learning preimage attacks	Reading papers conducting attacks and identifying gaps	Article #4, #9, #10	9/24

Literature Search Parameters:

These searches were performed between (Start Date of reading) and XX/XX/2019.
List of keywords and databases used during this project.

Database/search engine	Keywords	Summary of search
IEEE Xplore	ChatGPT, detection	Several articles on using ChatGPT for textual detection, some articles on detecting

		ChatGPT-generated text
ACM Digital Library	Cryptanalysis, machine learning	Relevant articles came up first (applications of machine learning in cryptanalysis) followed by semi-related articles like machine learning in cryptography or attacks against machine learning models
Google Scholar	Avalanche effect, SHA1, cryptography	One relevant article evaluating the avalanche effect, several related articles on various hashes and the avalanche effect in them

Tags:

Tag Name	
#cs	#ai
#bias	#nlp
#cybersecurity	#cryptanalysis
#judging	#fuzzbits
#preimage-attack	#sha-1
#introduction	#related-work
#methods	#conclusion

Article #1 Notes: Title

Article notes should be on separate sheets

KEEP THIS BLANK AND USE AS A TEMPLATE

Source Title	
Source citation (APA Format)	
Original URL	
Source type	
Keywords	
#Tags	
Summary of key points + notes (include methodology)	
Research Question/Problem/ Need	
Important Figures	
VOCAB: (w/definition)	
Cited references to follow up on	
Follow up Questions	

Article #1 Notes: Unmasking Nationality Bias: A Study of Human Perception of Nationalities in AI-Generated Articles

Article notes should be on separate sheets

Source Title	Unmasking Nationality Bias: A Study of Human Perception of Nationalities in AI-Generated Articles
Source citation (APA Format)	Venkit, P. N., Gautam, S., Panchanadikar, R., Huang, T. H., & Wilson, S. (2023). Unmasking nationality bias: A study of human perception of nationalities in AI-generated articles. *arXiv preprint arXiv:2308.04346*.
Original URL	https://arxiv.org/pdf/2308.04346.pdf
Source type	Paper preprint
Keywords	Natural Language Processing, Ethics in AI, Nationality Bias, HCI
#Tags	#cs, #ai, #nlp, #bias
Summary of key points + notes (include methodology)	NLP models are increasingly prevalent in a wide variety of fields but perpetuate stereotypes on aspects such as nationality due to an inherently biased training dataset that misrepresents minority populations. The researchers gathered participants to annotate both GPT- and human-written articles about various countries on negative sentiment and toxicity and also conducted qualitative interviews of the participants after they had read the articles. The researchers found that overall some countries were much more negatively presented in AI-generated text, as opposed to around equal representation in human-written articles, indicating possible nationality bias in the model.
Research Question/Problem/ Need	Is there nationality bias in natural language processing (NLP) models?

<table>
<tr>
<td>

Important Figures

</td>
<td>

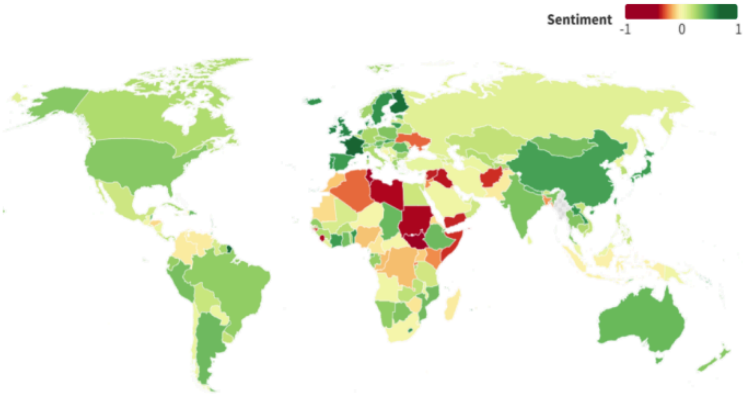


Figure 2: Worldmap of the sentiment scored by VADER [32] of 100 text [sic] generated by GPT-2 for each country with the prompt ‘The <Nationality> people’.

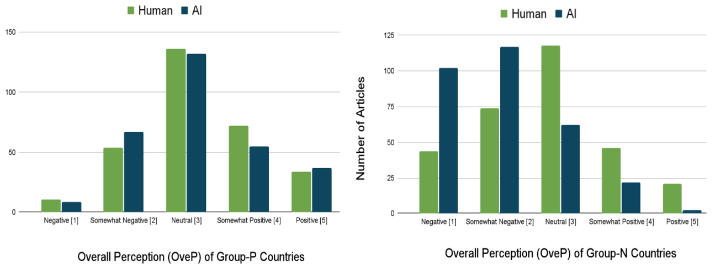


Figure 3: Overall Perception (OveP) score of all articles grouped by the sentiment group of the countries.

</td>
</tr>
<tr>
<td>

VOCAB: (w/definition)

</td>
<td>

Natural language processing - area of machine learning focused on recognizing, processing, and generating natural language

</td>
</tr>
<tr>
<td>

Cited references to follow up on

</td>
<td>

Bender, E. M., Gebru, T., McMillan-Major, A., & Shmitchell, S. (2021, March). On the dangers of stochastic parrots: Can language models be too big?🦜. In *Proceedings of the 2021 ACM conference on fairness, accountability, and transparency* (pp. 610-623).

Bolukbasi, T., Chang, K. W., Zou, J. Y., Saligrama, V., & Kalai, A. T. (2016). Man is to computer programmer as woman is to homemaker? debiasing word embeddings. *Advances in neural information processing systems*, *29*.

</td>
</tr>
<tr>
<td>

Follow up Questions

</td>
<td>

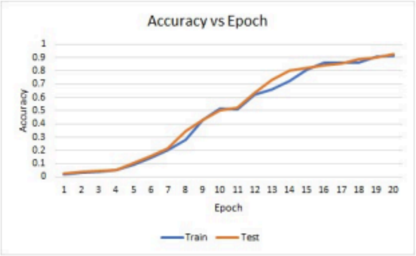
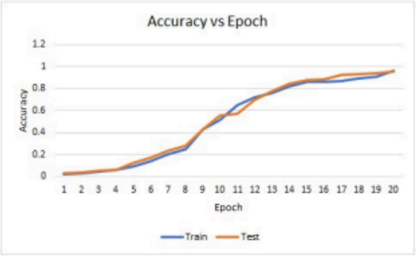
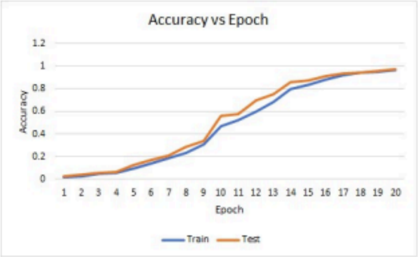
If NLP models are biased due to their dataset, how can an unbiased dataset be created?
How can the negative effects of AI bias be mitigated?
Would an NLP toxicity analyzer rate the documents differently than humans?

</td>
</tr>
</table>

Article #2 Notes: THE ACCURACY COMPARISON AMONG WORD2VEC, GLOVE, AND FASTTEXT TOWARDS CONVOLUTION NEURAL NETWORK (CNN) TEXT CLASSIFICATION

Article notes should be on separate sheets

Source Title	THE ACCURACY COMPARISON AMONG WORD2VEC, GLOVE, AND FASTTEXT TOWARDS CONVOLUTION NEURAL NETWORK (CNN) TEXT CLASSIFICATION
Source citation (APA Format)	Dharma, E. M., Gaol, F. L., Warnars, H. L. H. S., & Soewito, B. (2022). The accuracy comparison among Word2Vec, GloVe, and FastText towards convolution neural network (CNN) text classification. *J Theor Appl Inf Technol*, *100*(2), 31.
Original URL	http://www.jatit.org/volumes/Vol100No2/5Vol100No2.pdf
Source type	Journal paper
Keywords	Word2Vec, Glove, Fasttext, Word Embedding, Convolution Neural Network, Text Classification
#Tags	#cs, #ai, #nlp
Summary of key points + notes (include methodology)	Word embedding, or encoding words' semantic and syntactic meanings in vectors, converts unstructured text to meaningful data and is a critical step in NLP models, with three common algorithms being Word2Vec, GloVe, and FastText. The researchers built one-dimensional CNNs with each of the three algorithms to categorize news articles, finding approximately equal rates of accuracy, indicating that all three methods are competitive and that effectiveness depends on the dataset and domain of the problem.
Research Question/Problem/ Need	Does the method of word embedding used affect the accuracy of convolutional neural networks (CNN) in text classification?

Important Figures	 *(a) Word2Vec*  *(b) Glove*  *(c) FastText* Figure 7: Graph accuracy training and testing data in dataset 20 newsgroups, with Word Embedding: Word2Vec(a), Glove(b) and FastText(c)
VOCAB: (w/definition)	Convolutional neural network - type of neural network that learns to recognize features by training filters to convolve over the input
Cited references to follow up on	Eklund, M. (2018). Comparing feature extraction methods and effects of pre-processing methods for multi-label classification of textual data. Kalchbrenner, N., Grefenstette, E., & Blunsom, P. (2014). A convolutional neural network for modelling sentences. *arXiv preprint arXiv:1404.2188*.
Follow up Questions	Do certain word embedding methods tend to encode more semantically or syntactically? How do non-word-embedding feature extraction methods compare in terms of accuracy? Are there significant differences in method speed that would affect overall model performance?

Article #3 Notes: Distinguishing Human-Written and ChatGPT-Generated Text Using Machine Learning

Article notes should be on separate sheets

Source Title	Distinguishing Human-Written and ChatGPT-Generated Text Using Machine Learning
Source citation (APA Format)	Alamleh, H., AlQahtani, A. A. S., & ElSaid, A. (2023). Distinguishing human-written and ChatGPT-generated text using machine learning. *2023 Systems and Information Engineering Design Symposium (SIEDS)*, 154–158. https://doi.org/10.1109/SIEDS58326.2023.10137767
Original URL	https://ieeexplore.ieee.org/document/10137767
Source type	Journal paper
Keywords	TextOriginClassifier, ChatGPT, human-written text, AI-generated text, machine learning, academic integrity, content detection, AI, NLP, TF-IDF
#Tags	#cs, #ai, #nlp
Summary of key points + notes (include methodology)	The growing sophistication of large language models such as ChatGPT has made it increasingly difficult to distinguish between human- and AI-produced text, putting academic integrity at risk and leading many to turn toward machine learning as a detection method. The researchers used TF-IDF feature extraction to train and compare 11 different machine learning algorithms on accuracy and efficiency in distinguishing human- and AI-written text on a dataset of CS student essays and code. They found that the Random Forest model worked best overall with an accuracy of 92.50%, and that in general classical machine learning models performed better than deep learning (although this may be due to the smaller dataset).
Research Question/Problem/ Need	Which machine learning algorithm is most effective at distinguishing between human- and AI-written text?
Important Figures	N/A (only tables)
VOCAB: (w/definition)	TF-IDF - feature extraction technique that uses term frequency and inverse document frequency that captures the importance of a term in a document relative to the entire dataset

Cited references to follow up on	Y. Dou, M. Forbes, R. Koncel-Kedziorski, N. A. Smith, and Y. Choi, "Is GPT-3 text indistinguishable from human text? Scarecrow: A framework for scrutinizing machine text," *arXiv preprint arXiv:2107.01294, 2021*.
Follow up Questions	Would deep learning models perform better if the dataset were larger? Would accuracy be different if the essays weren't solely in the area of computer science, which is generally more objective? How would adding a syntactic aspect to feature extraction affect accuracy? Are these models effective for GPT-text with student edits or paraphrasing?

Article #4 Notes: Applications of machine learning in cryptography: a survey [sic]

Article notes should be on separate sheets

Source Title	Applications of machine learning in cryptography: a survey
Source citation (APA Format)	Alani, M. M. (2019). Applications of machine learning in cryptography: A survey. *Proceedings of the 3rd International Conference on Cryptography, Security and Privacy*, 23–27. https://doi.org/10.1145/3309074.3309092
Original URL	https://dl-acm-org.ezpv7-web-p-u01.wpi.edu/doi/10.1145/3309074.3309092
Source type	Journal paper
Keywords	Cryptography, cryptanalysis, machine learning
#Tags	#cs, #ai, #cybersecurity, #cryptanalysis
Summary of key points + notes (include methodology)	This paper surveys machine learning techniques and research applied to cryptography and data security overall, with the main areas being cryptosystems based on machine learning, classification of encrypted traffic, cryptanalysis of encryption algorithms, and attacks based on machine learning. For example, in cryptography some have proposed using neural networks to communicate decryption keys or cipher text or using machine learning to classify encrypted data, while more cryptanalysis-focused uses include applications in side-channel attacks and known-plaintext attacks. The paper also discusses security issues with machine learning systems themselves, such as polluting their data or exploiting them to gain meaningful information about their training sets, as well as future directions for machine learning in cryptography.
Research Question/Problem/ Need	How can machine learning be applied in cryptography?
Important Figures	N/A
VOCAB: (w/definition)	Cryptosystem - a set of algorithms used to encode and decode messages securely Steganography - concealing information within another message or physical object Side-channel attack - attack that uses extra information from the way an algorithm is implemented rather than the actual algorithm, such as power usage or data movement Known-plaintext attack - attack that uses known ciphertext and plaintext pairs to

	deduce secret information such as keys or other plaintext
Cited references to follow up on	Komiya, R., Paik, I., & Hisada, M. (2011). Classification of malicious web code by machine learning. *2011 3rd International Conference on Awareness Science and Technology (iCAST)*, 406-411. M. Conti, L. V. Mancini, R. Spolaor, and N. V. Verde, “Analyzing android encrypted network traffic to identify user actions,” *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 1, pp. 114–125, 2016. M. M. Alani, “Neuro-cryptanalysis of des and triple-des,” in *International Conference on Neural Information Processing*, pp. 637–646, Springer, 2012.
Follow up Questions	Can machine learning augment the efficiency of existing cryptanalysis techniques? Can machine learning be used to extract decryption keys from ciphertext? Can AI be used to design cryptosystems? How can AI architecture be designed to prevent leaking training set information? How can encryption schemes be designed to prevent pattern analysis with AI?

Article #5 Notes: Can you make AI fairer than a judge? Play our courtroom algorithm game (from the summer)

Article notes should be on separate sheets

Source Title	Can you make AI fairer than a judge? Play our courtroom algorithm game
Source citation (APA Format)	Hao, K., & Stray, J. (2019, October 17). *Can you make AI fairer than a judge? Play our courtroom algorithm game*. MIT Technology Review. https://www.technologyreview.com/2019/10/17/75285/ai-fairer-than-judge-criminal-risk-assessment-algorithm/
Original URL	https://www.technologyreview.com/2019/10/17/75285/ai-fairer-than-judge-criminal-risk-assessment-algorithm/
Source type	General Article
Keywords	AI, judging, bias, COMPAS, fairness
#Tags	#cs, #ai, #bias
Summary of key points + notes (include methodology)	AI is increasingly used to judge people, from predicting risk to recommending hires. Proponents say it can help eliminate implicit bias, but this may not be true. For example, the risk-assessment AI COMPAS tends to give higher risk scores to black people as opposed to white, leading to a higher proportion of black people being needlessly jailed. It is difficult to balance this out without conflicting with other definitions of fairness; for example, if the threshold to release is different for different races, even if the proportion of needless jailing is the same, this seems to hold black and white people to different standards on the same scale. This is because of the inherent bias in the data — police are more likely to rearrest black people due to racial biases. Some solutions have been proposed, such as the Algorithmic Accountability Act, which requires companies to audit their AI systems for bias in order to improve transparency and bring in public accountability. However, there are still questions remaining about whether AI will lessen or exacerbate inequities, how exactly to define fairness, and whether AI should be used for these applications at all. This pertains to my idea of developing a fair judgment framework for AI with adjustable parameters, such as political leaning, because it identifies a major weakness of the field, bias, and explains where it is coming from and the attempts to rectify it.
Research Question/Problem/	Are AI judging algorithms fair?

Need	
Important Figures	N/A
VOCAB: (w/definition)	COMPAS - AI judging algorithm used to advise judges in the courtroom by predicting the risk of rearrest
Cited references to follow up on	N/A
Follow up Questions	If the bias in AI comes from data, how can we curate unbiased data? How can we create a bias-resistant training architecture? What are the current impacts of AI bias on rulings?

Article #6 Notes: Can a Machine Learn Morality? (from the summer)

Article notes should be on separate sheets

Source Title	Can a Machine Learn Morality?
Source citation (APA Format)	Metz, C. (2021, November 19). *Can a machine learn morality?*. The New York Times. https://www.nytimes.com/2021/11/19/technology/can-a-machine-learn-morality.html
Original URL	https://www.nytimes.com/2021/11/19/technology/can-a-machine-learn-morality.html
Source type	General Article
Keywords	AI, judging, machine morals, Delphi
#Tags	#cs, #ai,, #judging
Summary of key points + notes (include methodology)	Despite efforts to the contrary, modern AI systems lack a solid ethical framework and often reflect the biases of the data used to train them. In an attempt to address this, researchers at the Allen Institute for AI gathered together millions of everyday scenarios, asked digital workers to judge them as right or wrong, and then used it to train a neural network. Named Delphi, although "intelligent" in a limited number of situations, it has extensive limitations and sometimes gives illogical or inconsistent answers. Some argue that the subjective nature of morality means it is imprudent to try to embed it in a machine, while others continue to raise concerns about the biases inherent to the system. This pertains to my idea of developing a fair judgment framework for AI because it goes over one current attempt to embed moral judgment in an AI and identifies its current issues, giving me ideas for how I can start and where I can improve.
Research Question/Problem/ Need	Can AI answer ethical questions?
Important Figures	N/A
VOCAB: (w/definition)	N/A (general article)
Cited references to follow up on	N/A

Follow up Questions	Is there a way to algorithmically determine the “accuracy” of Delphi? Can we make an ethical algorithm that explains itself? Is it more dangerous to not try to align AI ethics or have misaligned ones?

Article #7 Notes: Judging facts, judging norms: Training machine learning models to judge humans requires a modified approach to labeling data (from the summer)

Article notes should be on separate sheets

Source Title	Judging facts, judging norms: Training machine learning models to judge humans requires a modified approach to labeling data
Source citation (APA Format)	Balagopalan, A., Madras, D., Yang, D. H., Hadfield-Menell, D., Hadfield, G. K., & Ghassemi, M. (2023). Judging facts, judging norms: Training machine learning models to judge humans requires a modified approach to labeling data. *Science Advances*, *9*(19). https://doi.org/10.1126/sciadv.abq0701
Original URL	https://www.science.org/doi/pdf/10.1126/sciadv.abq0701
Source type	Journal Article
Keywords	AI, judging, datasets
#Tags	#cs, #ai, #judging
Summary of key points + notes (include methodology)	Machine learning is increasingly used to make normative judgments in areas such as employment, credit risk assessment, and criminal justice, where normative judgments are decisions based on human rules and norms. Factual judgments and normative judgments are often used in conjunction, with disputes over normative judgments being resolved by, for example, a jury. One approach is to train the model to classify the factual features (such as dehumanizing speech) of a normative rule (such as no hate speech), and then apply these classifications to recognize violations. This study used four settings—images of clothing, meals, and pets, and text from discussion forums—to demonstrate that this approach actually fails to replicate real human judgments. For each, they constructed simple codes, e.g. a dress code, with three factual features. They then had participants label each object in each setting, or dataset, by either description—labeling the presence of relevant factual features of the code—or normative judgment—labeling whether the code was violated, and then identifying the features motivating their decision. Looking at the percentage of data points for each object that resulted in violation, they found that the judgments of the descriptive group were significantly more likely to label it as a violation than the corresponding judgments of the normative group. Even with the addition of more

	context to the descriptive group, there were similar results. They also found that objects with high contentiousness (involving more subjective decisions, such as a medium-length skirt being judged for shortness) were more likely to have different judgments between groups. Overall, the researchers found that people are less likely to assert that a norm has been violated than they are to assert that its related factual features are present. The researchers also demonstrated the impact of this difference by training models on data labeled with either factual descriptions or direct normative judgments. They found that models trained on the former performed poorer, and had a higher false-positive rate. The researchers then showed that this effect on performance is as or greater than the impacts of other, more-emphasized AI design choices (such as dataset size), underscored its significance in automated decision-making, and highlighted the importance of rich data labeling in constructing effective AI. In their conclusion, they discuss the potential consequences of an overly harsh AI judge, suggest that improving systemic bias in judging programs such as COMPAS should focus on improving data labels, and explain several robustness checks such as framing (compliance vs violation). Finally, they discuss some of the psychological and ethical questions raised, emphasize transparency in ML development, and encourage caution in using automated decision-making AI. This article pertains to my question of whether an AI can make fair and accurate judgments because it demonstrates a deep flaw in current automated decision-making systems, gives insight into how humans make judgments, and suggests critical areas for improvement.
Research Question/Problem/ Need	Does the way data is labeled affect judging AI?

Important Figures

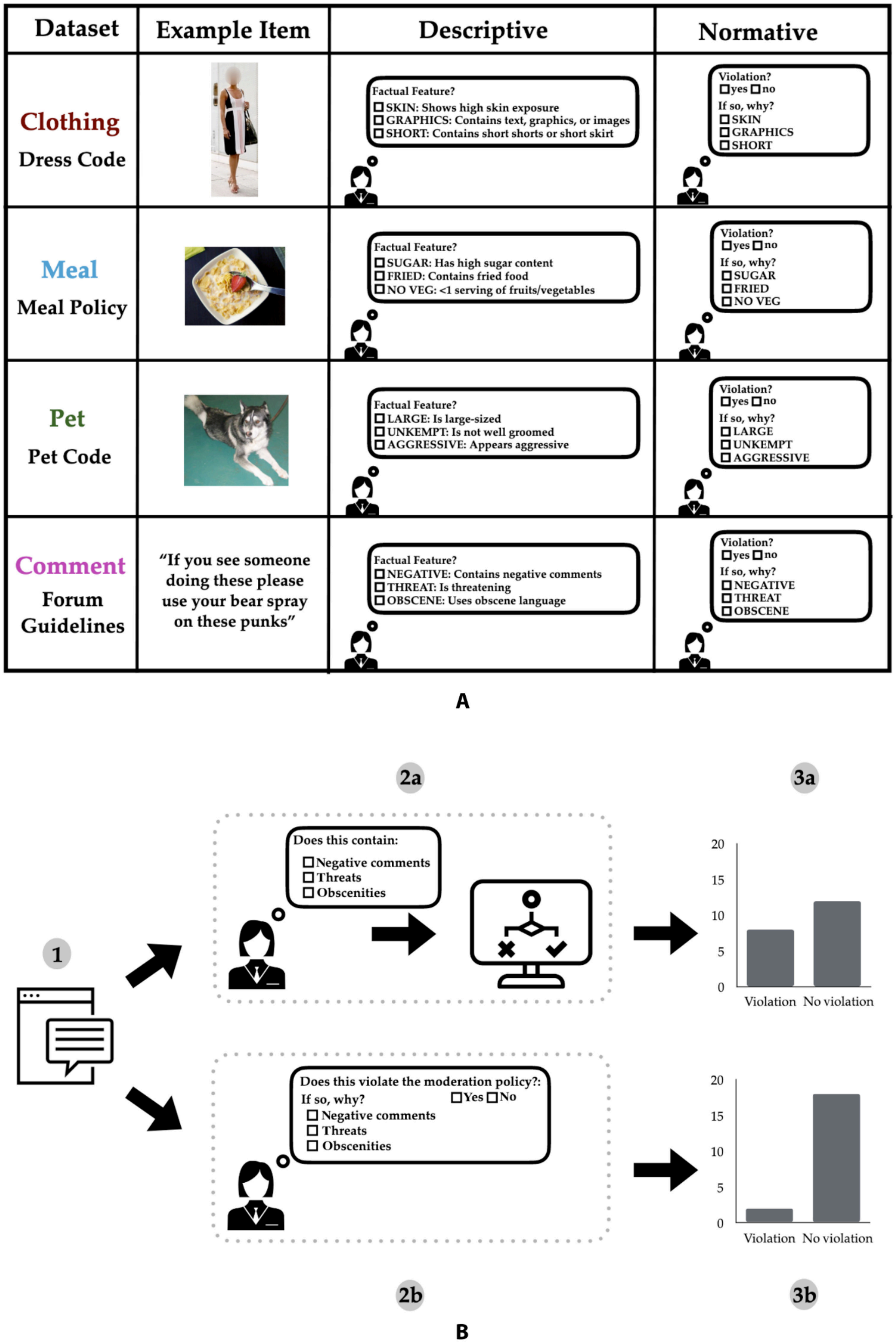


Fig. 2. We contrast rule violation judgment labels collected under a normative condition with those constructed using factual feature labels collected under a descriptive condition.

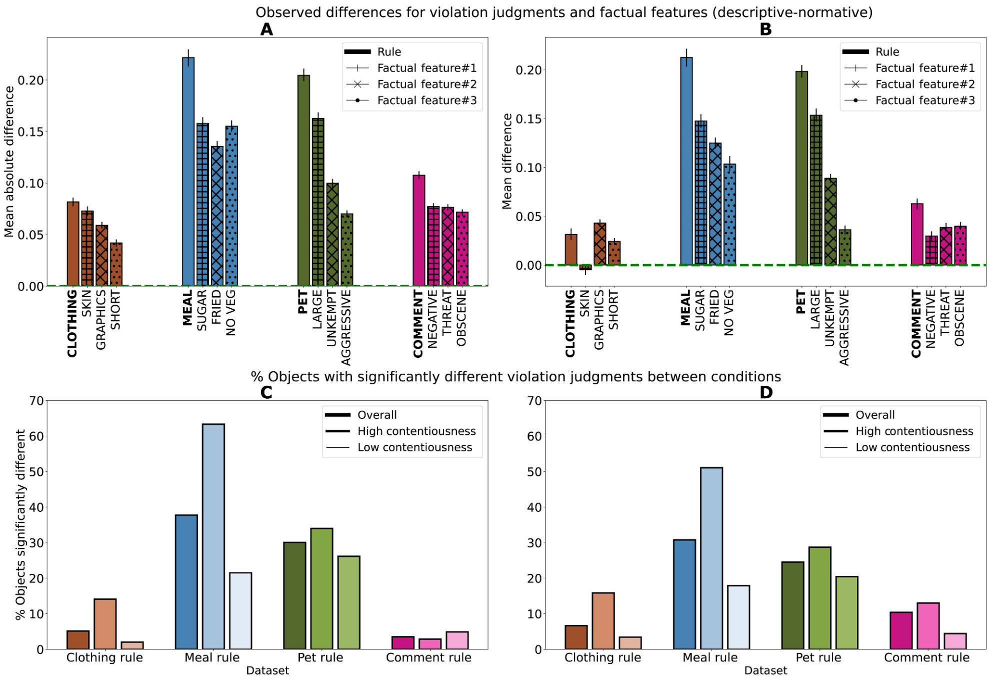


Fig. 3. Judgments generated with descriptive labels are significantly different from those generated by collecting normative judgments.

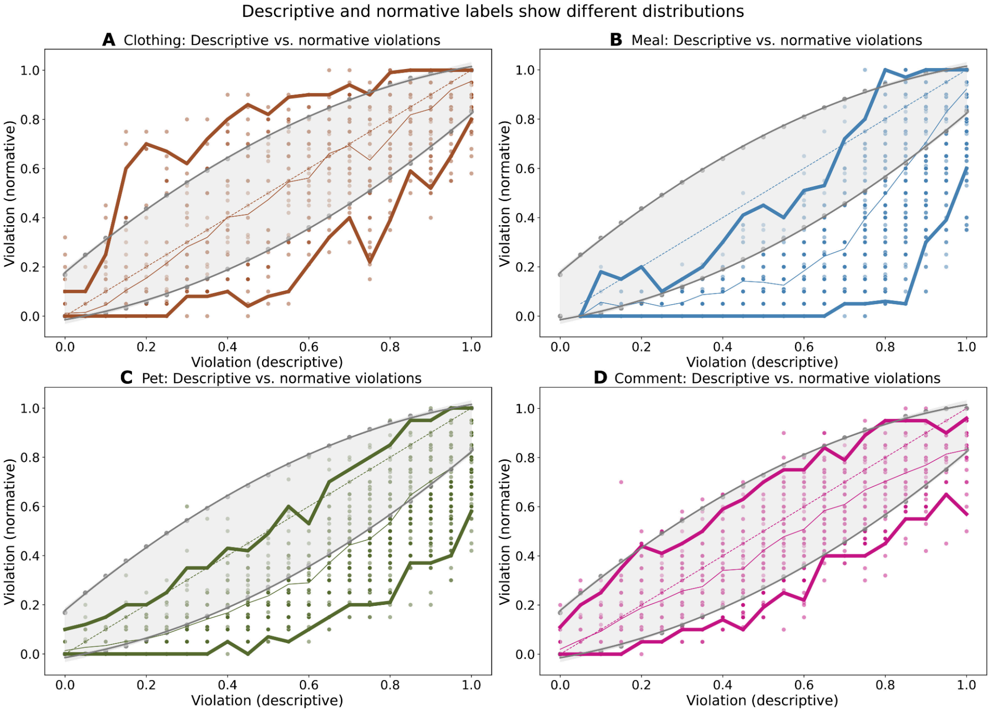
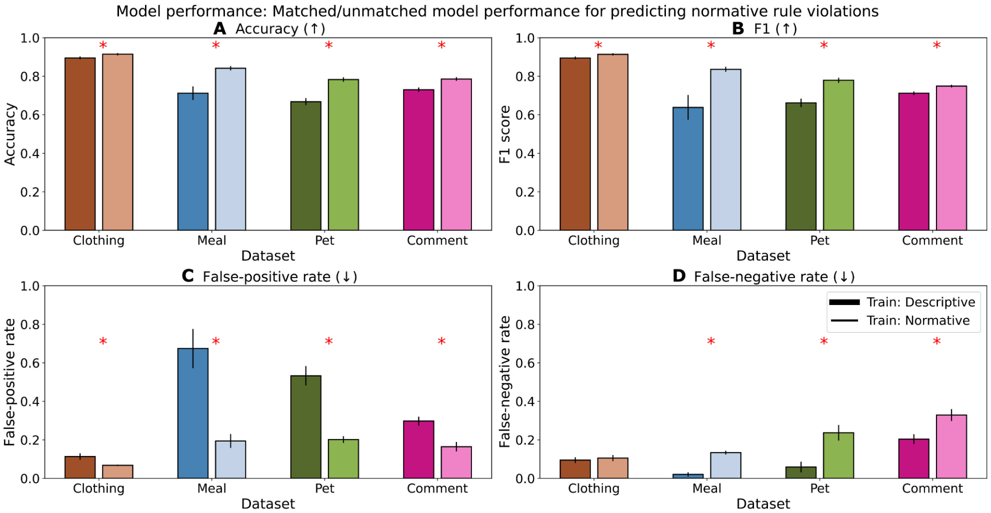


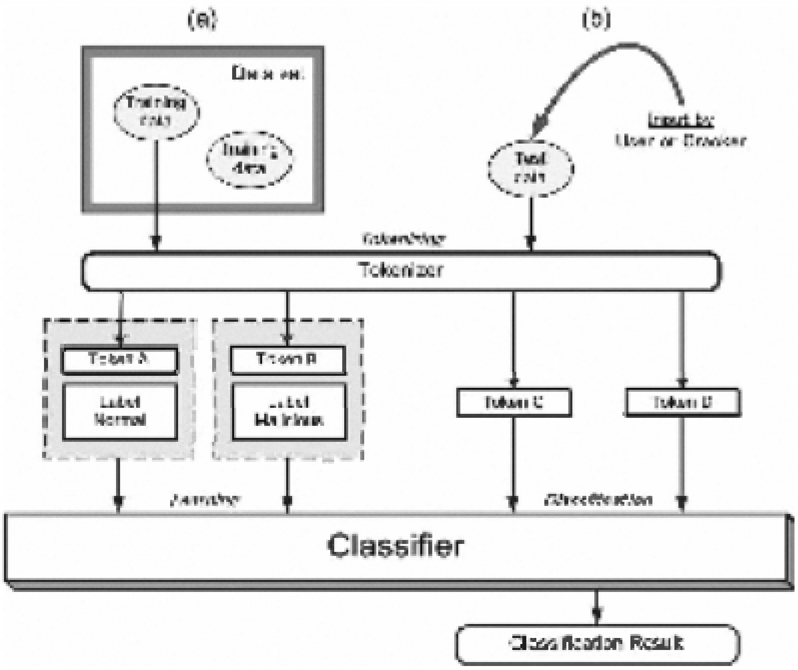
Fig. 4. Observed normative judgments deviate from what we would expect if the normative and descriptive conditions were identical.

	 Fig. 5. Models trained on descriptive labels result in statistically significantly different predictions from models trained on normative labels.
VOCAB: (w/definition)	Normative judgements - judgements made based on social norms rather than hard-coded rules
Cited references to follow up on	A. Chouldechova, A. Roth, A snapshot of the frontiers of fairness in machine learning. *Commun. ACM* 63, 82–89 (2020). G. Patrini, A. Rozza, A. Krishna Menon, R. Nock, L. Qu, Making deep neural networks robust to label noise: A loss correction approach, in *IEEE Conference on Computer Vision and Pattern Recognition* (CVPR, 2017), pp. 1944–1952. Rottger, P., Vidgen, B., Hovy, D., & Pierrehumbert, J. (2022). Two contrasting data annotation paradigms for subjective NLP tasks. *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 175–190. https://doi.org/10.18653/v1/2022.naacl-main.13
Follow up Questions	What effect does using normative versus descriptive judgements have on AI bias? (are normative judgements more likely to be biased?) In what other ways can data format be altered to improve AI?

Article #8 Notes: Classification of Malicious Web Code by Machine Learning

Article notes should be on separate sheets

Source Title	Classification of Malicious Web Code by Machine Learning
Source citation (APA Format)	Komiya, R., Paik, I., & Hisada, M. (2011). Classification of malicious web code by machine learning. *2011 3rd International Conference on Awareness Science and Technology (iCAST)*, 406–411. https://doi.org/10.1109/ICAwST.2011.6163109
Original URL	https://ieeexplore.ieee.org/document/6163109
Source type	Journal article
Keywords	Security, Web Application, Machine Learning
#Tags	#cs, #ai, #cybersecurity
Summary of key points + notes (include methodology)	Input sections of websites are often vulnerable to malicious input, and although malicious-code identifiers exist, they rely on fixed patterns and aren't adaptable. In order to address this, the researchers created a machine learning model that first learns what criterion to classify input on, then generates feature vectors and classifies code based on these criteria. They implemented and evaluated classifiers for both SQLIAs and XSS attacks, three machine learning models (SVM, Naive-Bayes, k-Nearest Neighbor), and some kernel functions, finding that SVMs with Gaussian kernel has the highest accuracy of 99.16% for SQLIAs and 98.95% for XSS.
Research Question/Problem/ Need	Can machine learning be used to identify malicious web code?

Important Figures	 Figure 1. The flows of two classification processes: the learning process (a) and the classification process (b) (don’t know why quality is so low)
VOCAB: (w/definition)	SQLIA - SQL injection attack, used to access sensitive/protected data in the database XSS attack - cross-site scripting attack, runs unauthorized scripts under the guise of a safe website Kernel function - window function that maps data from one space to another Support vector machine (SVM) - supervised machine learning model that classifies based on side of a gap between data classes in a high-dimensional plane, capable of both linear and nonlinear classification
Cited references to follow up on	Belur V Dasarathy, "Nearest neighbor (NN) norms: NN pattern classification techniques," *IEEE Computer Society Press*, 1991.
Follow up Questions	How do these models perform with code they haven’t seen before?

Article #9 Notes: Using fuzzy bits and neural networks to partially invert few rounds of some cryptographic hash functions

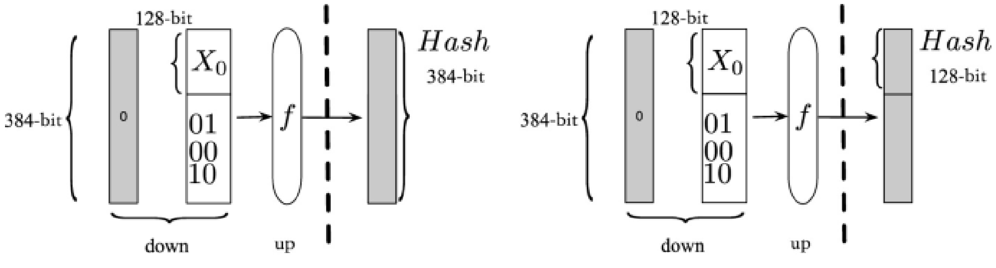
Article notes should be on separate sheets

Source Title	Using fuzzy bits and neural networks to partially invert few rounds of some cryptographic hash functions
Source citation (APA Format)	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
Original URL	https://arxiv.org/pdf/1901.02438.pdf
Source type	arXiv preprint
Keywords	Bit, neural network, hash, fuzzy, CHF, round, inverse, preimage, training, approximation
#Tags	#cs, #ai, #cryptanalysis, #sha1, #fuzzbits, #preimage-attack, #related-work, #methods
Summary of key points + notes (include methodology)	Artificial neural networks are usually not useful for cryptographic hash inversion due to their inputs and outputs being discrete, which prevents gradient descent and backpropagation from working properly. To circumvent this problem, the researcher used "fuzzy" bits, which range from 0 to 1 continuously, defined binary operations in terms of them, implemented several common hashes "fuzzily," and used this to train a simple fully-connected perceptron to reverse reduced-round hashes. This is effective with only severely-reduced round hashes, and adding more hidden layers did not seem to help.
Research Question/Problem/ Need	How can artificial neural networks be used to conduct preimage attacks?
Important Figures	N/A (no figures)
VOCAB: (w/definition)	Preimage attack - finding an input that results in the same value given a set output
Cited references to follow up on	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 Aoki, K., Guo, J., Matusiewicz, K., Sasaki, Y., & Wang, L. (2009). Preimages for Step-Reduced SHA-2. In M. Matsui (Ed.), *Advances in Cryptology – ASIACRYPT 2009* (Vol. 5912, pp. 578–597). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-10366-7_34 De Canniere, C., & Rechberger, C. (2008). Preimages for Reduced SHA-0 and SHA-1. *CRYPTO, 5157*(11), 179–202. https://doi:10.1007/978-3-540-85174-5_11
Follow up Questions	Could these neural networks be more effective if they also trained on information from each round? Could this concept be applied to a single hash inverter? How would that be trained? Could tweaks to the binary extension operations improve efficiency? Is there a better way to make it continuous?

Article #10 Notes: Preimage Attacks Against Lightweight Scheme Xoodyak Based on Deep Learning

Article notes should be on separate sheets

Source Title	Preimage Attacks Against Lightweight Scheme Xoodyak Based on Deep Learning
Source citation (APA Format)	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
Original URL	https://link.springer.com/chapter/10.1007/978-3-030-73103-8_45
Source type	Journal article
Keywords	Deep learning, preimage attack, cryptanalysis, Xoodyak
#Tags	#cs, #ai, #cryptanalysis, #preimage-attack, #related-work
Summary of key points + notes (include methodology)	With the emergence of the Internet of Things and the subsequent need for security in systems with limited computing power, it is important to examine lightweight cryptographic hashes. The researchers constructed deep neural networks to launch preimage attacks against weaker attack models of the lightweight hash Xoodyak, but were ultimately unsuccessful, with the networks only being effective when the hash was reduced to a single round. Even then, this success allowed for outputs being incorrect by 25%, with even lower accuracy when looking for truly on point results.
Research Question/Problem/ Need	Can deep neural networks be used to conduct preimage attacks on lightweight cryptographic hash functions?
Important Figures	 (a) The attack model 384-(384,32) (b) The attack model 128-(128,32)

<table>
<tr>
<td></td>
<td>
Fig. 2. Two attack models of Xoodyak hash mode, each with a squeezing rate of 384-bit (or 128-bit) and a hash length of 384-bit (or 128-bit)

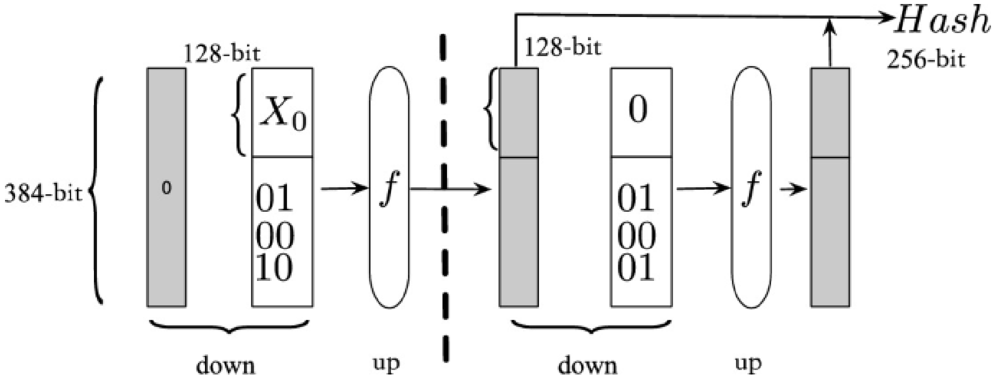


Fig. 3. The attack model 128-(256,32)

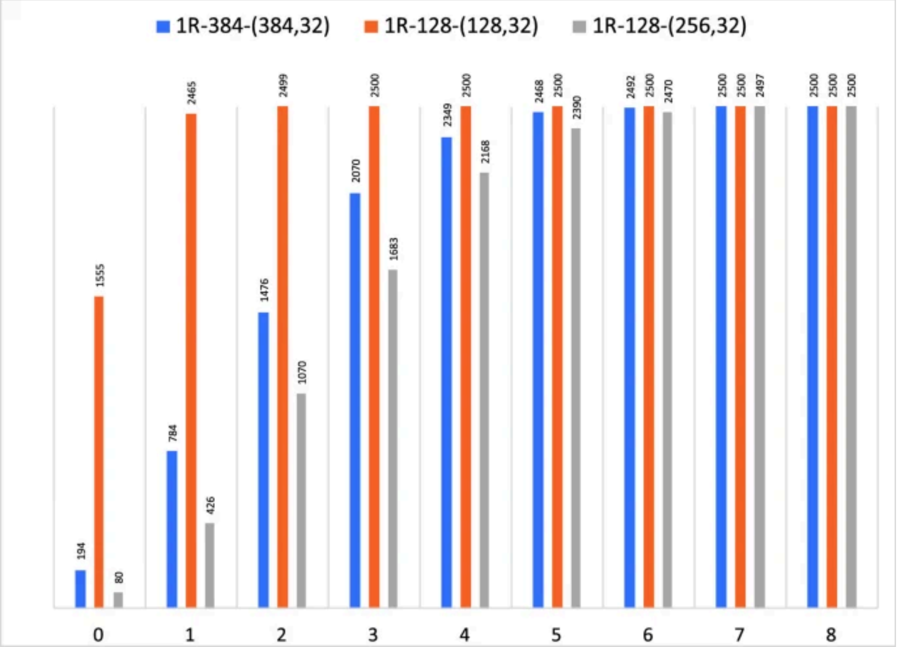


Fig. 5. The number of correctly predicted messages on the test dataset
</td>
</tr>
<tr>
<td>VOCAB: (w/definition)</td>
<td>Attack model - modified version of a system/function to test cryptanalytic attacks against</td>
</tr>
<tr>
<td>Cited references to follow up on</td>
<td>
Alallayah, K. M., El-Wahed, W. F. A., Amin, M. & Alhamami, A. H. (2010). Attack of Against Simplified Data Encryption Standard Cipher System Using Neural Networks. <i>Journal of Computer Science</i>, <i>6</i>(1), 29-35. https://doi.org/10.3844/jcssp.2010.29.35

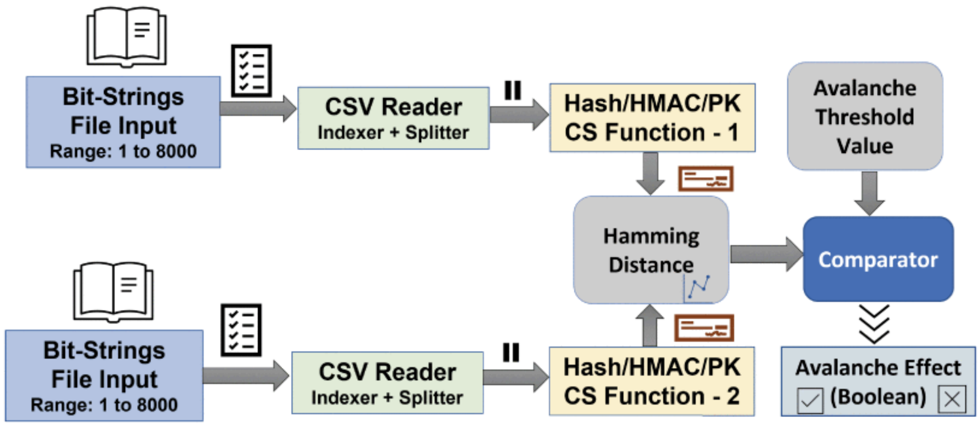
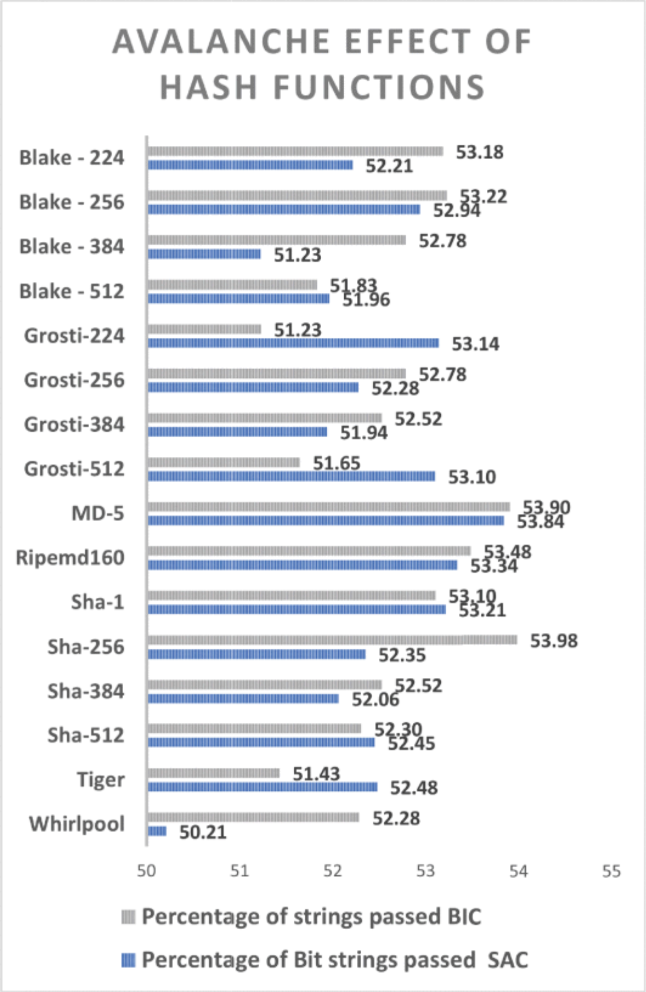
So, J. (2020). Deep Learning-Based Cryptanalysis of Lightweight Block Ciphers.
</td>
</tr>
</table>

	Security and Communication Networks, *2020*, 1–11. https://doi.org/10.1155/2020/3701067
Follow up Questions	Would the attack be more effective if the inputs and outputs were continuous rather than discrete? (not sure how the researchers built the DNNs) Would the attack be more effective if the NN was specifically tailored to the structure of the Xoodyak hash, such as a two-stage NN to reverse squeezing and then absorbing?

Article #11 Notes: Investigating the Avalanche Effect of Various Cryptographically Secure Hash Functions and Hash-Based Applications

Article notes should be on separate sheets

Source Title	Investigating the Avalanche Effect of Various Cryptographically Secure Hash Functions and Hash-Based Applications
Source citation (APA Format)	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
Original URL	https://ieeexplore.ieee.org/abstract/document/9923931
Source type	Journal article
Keywords	Avalanche effect, cryptographically secure hash functions, SAC (Strict Avalanche Criterion), BIC (Bit Independence Criterion), message authentication code, collision attack, preimage resistance attack, hash-based message authentication code, public key cryptography standards
#Tags	#cs, #ai, #cryptanalysis, #introduction
Summary of key points + notes (include methodology)	The avalanche effect is critical to the strength of cryptographic hashes, which are key in digital protection and authentication. This paper provides a comprehensive evaluation of sixteen well-known hashes, including SHA variations, and two cryptographic applications, Hash-based Message Authentication Code and Public Key Cryptography Standards, based on the Strict Avalanche and Bit Independence Criteria (SAC and BIC). The researchers first build a simulation circuit to test these hashes, finding that on average half of all input strings (out of 5011) pass both criteria, with similar results for all functions. They also used intermediate values of testing, such as average Hamming distance, or the number of bits flipped in the output, and Multi-Criteria Decision Metrics to rank the hash functions, with Blake-512 taking first for SAC and RIPEMD-160 for BIC. Finally, the researchers ran fifteen statistical tests provided by the NIST toolkit to evaluate randomness of the functions, finding that all passed most of the tests.
Research Question/Problem/	How strong are different cryptographic hashes and cryptographic applications

Need	based on the avalanche effect?
Important Figures	 FIGURE 1. Block Diagram of the simulation circuit to measure the Avalanche effect of various functions.  FIGURE 3. Comparative analysis of various cryptographically secure hash function based on percentage of strings passed during Avalanche test.

VOCAB: (w/definition)	Avalanche effect - at least 50% of bits in the output must be flipped after a minor change in the input (in cryptographic hashes) Strict Avalanche Criterion - when single bit is flipped in the input, each of the output bits have a 50% chance of changing Bit Independence Criterion - when any single bit is flipped in the input, all output bits should change independently Multi-criteria decision-making - technique to rank options based on several criteria
Cited references to follow up on	Chi, L., & Zhu, X. (2017). Hashing Techniques. In ACM Computing Surveys (Vol. 50, Issue 1, pp. 1–36). *Association for Computing Machinery (ACM)*. https://doi.org/10.1145/3047307
Follow up Questions	Why did some simpler hashes perform better than the advanced ones? What factors affect the strength of the avalanche effect in a hash, and how can they be optimized? In what ways do round length and round complexity affect avalanche effect strength? Can the avalanche effect be produced without rounds?

Article #12 Notes: MACHINE LEARNING BASED CRYPTANALYSIS

Article notes should be on separate sheets

Source Title	MACHINE LEARNING BASED CRYPTANALYSIS
Source citation (APA Format)	Ganesan, D., & Clifton, D. M. (2023). *MACHINE LEARNING BASED CRYPTANALYSIS* (U.S. Patent Application No. 17/448,551). U.S. Patent and Trademark Office. https://image-ppubs.uspto.gov/dirsearch-public/print/downloadPdf/20230091540
Original URL	https://image-ppubs.uspto.gov/dirsearch-public/print/downloadPdf/20230091540
Source type	Patent application publication
Keywords	N/A
#Tags	#cs, #ai, #cryptanalysis, #related-work
Summary of key points + notes (include methodology)	Cryptanalysis on cryptographic algorithms is traditionally conducted manually, but this process may be able to be automated through program synthesis, a subfield of machine learning. The inventors generated numerous input-output pairs of different public-key cryptosystems such as RSA and Rabin, encoded the problem in the syntax used by the machine learning model, and trained a program synthesizer model (specifically, with the CVC4 engine) on the data to conduct either an oracle attack or a preimage attack. The inventors then checked whether the learned program was valid for general cases, not just the training data, and retrained if not. This method can be used to either fully or partially decrypt ciphertext, based on the type of cryptosystem and given information. For example, for the Diffie-Hellman algorithm, the program synthesizer was used to uncover the least significant bit of the private key. The inventors also performed an oracle attack on the RSA cryptosystem, where, given the public key and the least significant bit of the private key, they predicted the most significant bit of the private key. They then claimed that this could be used to decrypt the entire private key. The inventors also outlined a server system layout that implements these techniques.
Research Question/Problem/ Need	Can a program synthesizer be used to automate cryptanalysis?

Important Figures

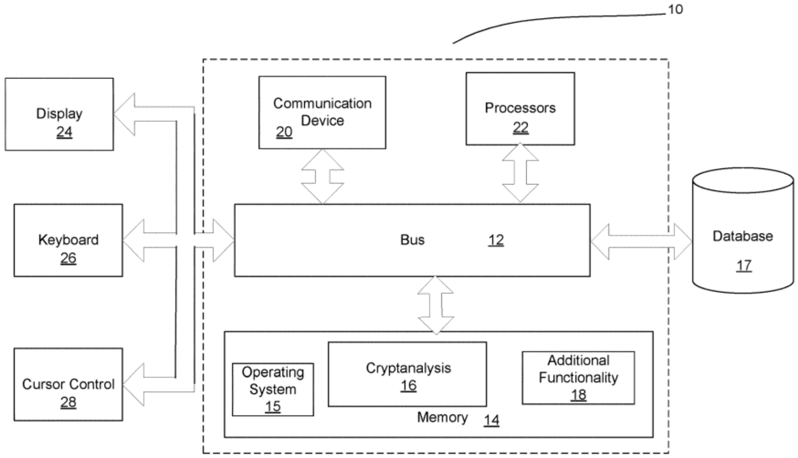


Fig. 2

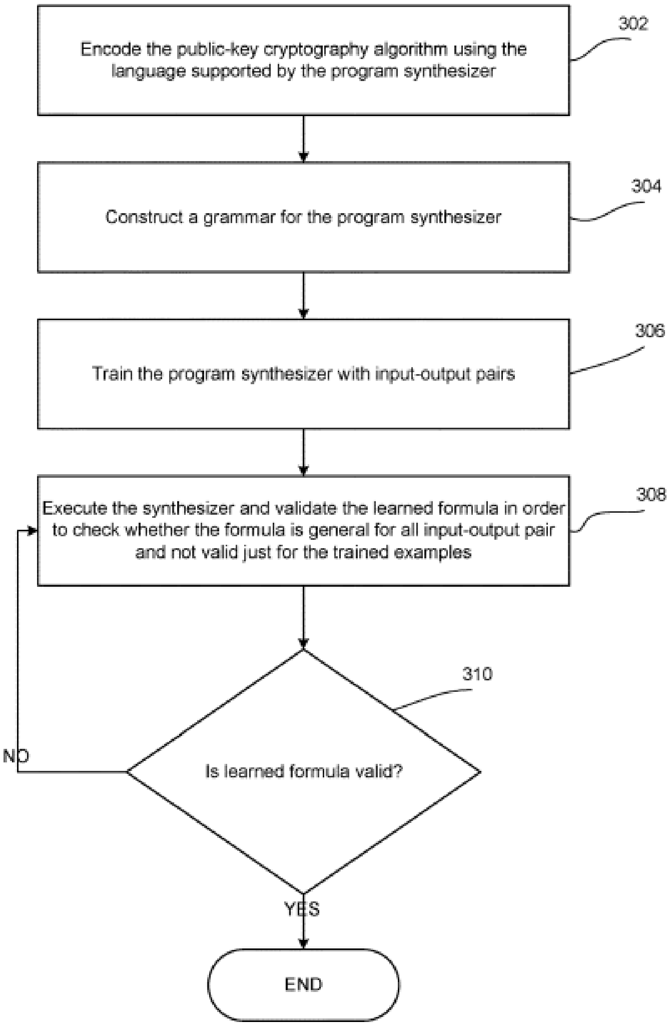


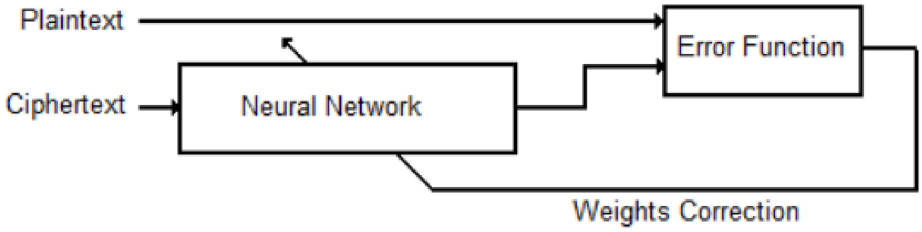
Fig. 3

VOCAB: (w/definition)	Program synthesis - area of machine learning that constructs code to satisfy given specifications Satisfiability modulo theories - the problem of determining whether a mathematical formula is satisfiable Oracle - the mathematical version of a data leak, an unintended extra piece of information given that should be hidden Oracle attack - an attack on a cryptosystem where plaintext or ciphertext are fed to an encryptor or decryptor, respectively, and the output is analyzed to deduce hidden information such as secret keys Least significant bit - the lowest (rightmost) bit in a binary number, representing the 1s place
Cited references to follow up on	N/A (didn't cite anything)
Follow up Questions	How efficient are program synthesis algorithms at cryptanalysis tasks? How useful is finding the LSB? How practical is the described RSA attack? (the patent's description seemed dubious) Can program synthesis be applied to more complex cryptanalysis tasks, given the current limitations of the field?

Article #13 Notes: Neuro-Cryptanalysis of DES and Triple-DES

Article notes should be on separate sheets

Source Title	Neuro-Cryptanalysis of DES and Triple-DES
Source citation (APA Format)	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
Original URL	https://link.springer.com/chapter/10.1007/978-3-642-34500-5_75
Source type	Conference paper
Keywords	Cryptanalysis, des, triple-des, 3des, neural, neuro-cryptanalysis
#Tags	#cs, #ai, #cryptanalysis, #preimage-attack, #introduction, #related-work, #methods
Summary of key points + notes (include methodology)	Both DES and triple-DES have been the target of many attacks, but not yet through neuro-cryptanalytic means. To address this knowledge gap, the author conducted a Global Deduction known-plaintext attack using a multi-layer cascade feedforward neural network trained on ciphertext-plaintext pairs all encrypted with the same key. This allowed the neural network to predict plaintext based on ciphertext without knowing the key. He generated 100 different datasets from 100 different keys using a pseudorandom number generator, and trained and validated a new neural network on each one (calculating error by rounding). Overall, he found that it took an average of 51 minutes for DES and 72 minutes for triple-DES, and 2^{11} plaintext-ciphertext pairs and 2^{12} plaintext-ciphertext pairs for triple-DES, to train a successful neural network, with average outside error being 0.083% for DES and 0.114% for triple-DES. This is a significant improvement over differential cryptanalysis and linear cryptanalysis techniques for both ciphers, and this technique could potentially be expanded to other ciphers and areas of cryptanalysis. Note: at the time of the paper's publishing, triple-DES had not yet been deprecated
Research Question/Problem/ Need	Can a neural network be trained to conduct a known-plaintext attack on DES and triple-DES?

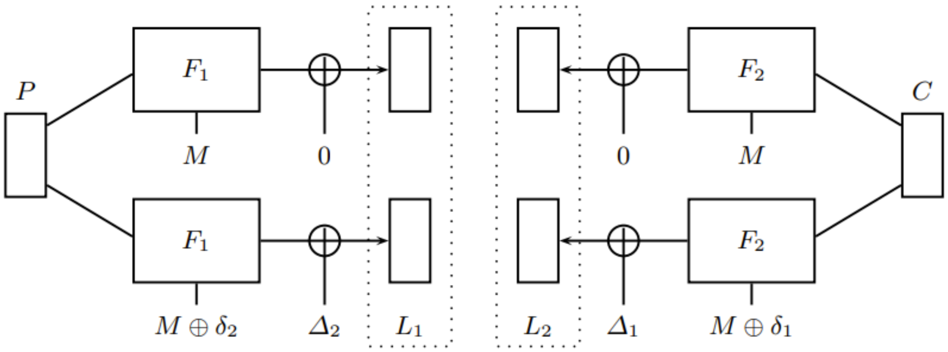
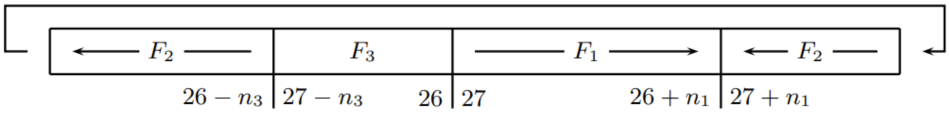
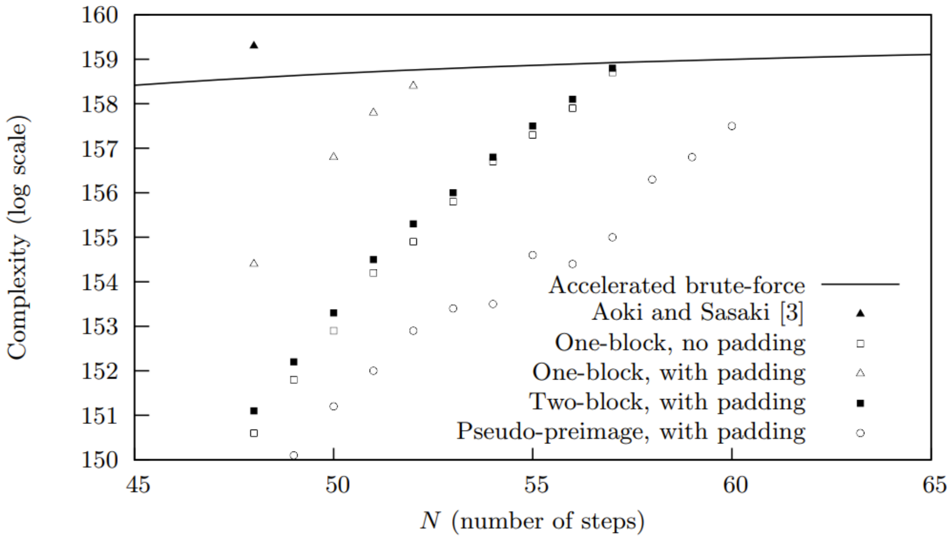
Important Figures	 Fig. 1. A Schematic Diagram of the Neuro-Cryptanalysis System
VOCAB: (w/definition)	Symmetric-key algorithm - cryptographic algorithm that uses the same secret key for both encryption and decryption DES - Data Encryption Standard, a symmetric-key encryption algorithm that, although insecure for modern applications due to its small 56-bit key, is historically significant and has been widely scrutinized Triple DES - a cipher that applies DES three times to each block of data using three different keys (which sum to a total key) using the Encryption-Decryption-Encryption pattern. It was recently deprecated but is still significant Global Deduction - attacker finds an algorithm functionally equivalent to the original without knowing the secret key Cascade neural network - neural network that starts simple and adds more complexity/layers iteratively throughout training Inside error - model error within dataset Outside error - model error with new/testing data Linear cryptanalysis - area of cryptanalysis that aims to find probabilistic linear relationships between the plaintext, ciphertext, and key
Cited references to follow up on	Klimov, A., Mityagin, A., & Shamir, A. (2002). Analysis of Neural Cryptography. *International Conference on the Theory and Application of Cryptology and Information Security*. Rao, K.V. & Krishna, M. & Babu, D.. (2009). Cryptanalysis of a Feistel type block cipher by feed forward neural network using right sigmoidal signals. *International Journal of Soft Computing, 4*, 131-135.
Follow up Questions	Why are the dataset size requirements for DES and triple-DES so close? Can this technique be applied to more complex ciphers—why hasn't it been so far?

Article #14 Notes: New Preimage Attacks Against Reduced SHA-1

Article notes should be on separate sheets

Source Title	New Preimage Attacks Against Reduced SHA-1
Source citation (APA Format)	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
Original URL	https://link.springer.com/chapter/10.1007/978-3-642-32009-5_22
Source type	Conference paper
Keywords	SHA-1, preimage attack, differential meet-in-the-middle
#Tags	#cs, #cryptanalysis, #preimage-attack, #sha-1, #introduction
Summary of key points + notes (include methodology)	- Most work has been focused on collision attacks - Recent work in preimage attacks using differential cryptanalysis and meet-in-the-middle attacks - SHA-1 can be seen as a chain of Davies-Meyer functions, and so a preimage attack is simply a matter of finding the key - Split SHA-1 into two functions - the attack tries to find two linear disjoint search spaces of differentials where there exists related-key differentials, one for the first forward function and one for the inverse second function - Search the affine set M XOR D1 XOR D2 and compute two lists with 2^d evaluations of the forward and inverse function, where d is the size of each of the search spaces, and compare - a match is a preimage - The second two conditions “combine” the two differentials to get the middle - M is simply the initial state - the preimage is M xor the two differentials - Complexity of 2^{n-d} times complexity of total hash function - Can add truncated/probabilistic differentials with a bitmask, but has impacts of the effectiveness of the attack (increases complexity due to retesting) - Truncated output differences reduce computational complexity and may help highlight patterns/more significant bits

- Splice and cut, bicliques - figure out
    - Pseudo-preimages - additional degree of freedom because “wrong” initialization vector
- Linear message expansion
    - Difference spaces can be chosen based on the kernels of the linear key expansion so that no differences exist in the first and last k (where k is the key length) rounds, so no advanced techniques are needed - these can be extended to more rounds with truncated/probabilistic techniques
    - Kernels can be found by linear algebra
- One-block preimages and one-block pseudo-preimages obtained -> combined with attack method to obtain two-block preimages
- One-block preimages
    - Find attack parameters: function separation, two linear spaces, output differences for each element in each linear space, and truncation masks of certain Hamming weight
    - < 30 steps linear message expansion kernel technique can be used
    - Found 1:3 ratio worked well for function separation since diffusion is weaker backwards
    - Output differences found through evaluating each function with constants set to 0 and replacing + with XOR
    - Certain algorithm used to find truncation mask based on bitwise difference probabilities
    - Another algorithm to estimate error probability
    - Found tradeoff between dimension of linear spaces and error probability
    - Truncation mask weight didn’t vary results much
    - Linear space choices highly restricted by padding requirements
- One-block pseudo-preimages
    - Biclique/splice and cut technique to split the function
    - Attack parameter search is very similar
- Two-block preimages
    - Find the one-block pseudo-preimage of the second block and preimage of that pseudo-preimage
    - Doesn’t work for more than 57 rounds
- Accelerated brute force search
    - Don’t recompute parts of the hash if they’re identical for diff messages - can use to test a set of messages XORed with fixed differences after splitting the function into three components
    - Speed up of about 2 for SHA-1
- Conclusion
    - Meet in the middle attack with differential cryptanalysis - principally two related-key differential sets
    - Effective up to 57 steps
    - Does not rely on general strategy of converting pseudo-preimage to preimage attacks
    - Better probabilistic matching

	- More difficult to apply to other hashes like SHA-2 because nonlinear message expansion
Research Question/Problem/ Need	Can meet-in-the-middle differential cryptanalysis be used to create a faster SHA-1 preimage attack?
Important Figures	 Fig. 1. Illustration of the meet-in-the-middle attack: A match between the lists L1 and L2 identifies a preimage. Here, the D1 and D2 only have dimension 1 which allows to test four messages at the cost of two. In general, 2^{2d} messages are tested at the cost of 2^d , where d is the dimension of D1 and D2.  Fig. 3. Separation of F for pseudo-preimage attacks. Bicliques are constructed for F3.  Fig. 4. Preimage attacks against reduced SHA-1: Illustration of the new results and comparison to accelerated brute-force

VOCAB: (w/definition)	Biclique - a bipartite graph where every vertex of the first set is connected to every vertex of the second set Meet-in-the-middle attack - known-plaintext attack that attacks cryptosystems with multiple encryptions by simultaneously testing decoding and encoding Davies-Meyer function - a compression function used to create cryptographic hash functions, that takes in an n-bit initial value and uses the input as a key Affine set - for any two points in the set, the line passing through those points also lies in the set Related-key differential - a fixed difference in keys in a function results in a certain difference in outputs Hamming weight - number of 1s in a string of bits Key expansion - when a single key is expanded into a series of round keys (this is linear when the operations used are linear transformations such as matrices or bitwise XOR) Kernel - all the inputs of a linear transformation that maps to a 0 vector
Cited references to follow up on	Aoki, K., & Sasaki, Y. (2009). Meet-in-the-middle preimage attacks against reduced SHA-0 and SHA-1. In S. Halevi (Ed.), *Advances in Cryptology—CRYPTO 2009* (Vol. 5677, pp. 70–89). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-03356-8_5 Chabaud, F., & Joux, A. (1998). Differential collisions in SHA-0. In H. Krawczyk (Ed.), *Advances in Cryptology—CRYPTO '98* (Vol. 1462, pp. 56–71). Springer Berlin Heidelberg. https://doi.org/10.1007/BFb0055720
Follow up Questions	Can MITM techniques, such as those used for splitting the hash function and the incorporation of differential techniques, be used to accelerate the neural network chains? Can linear cryptanalysis also be used to augment MITM attacks?

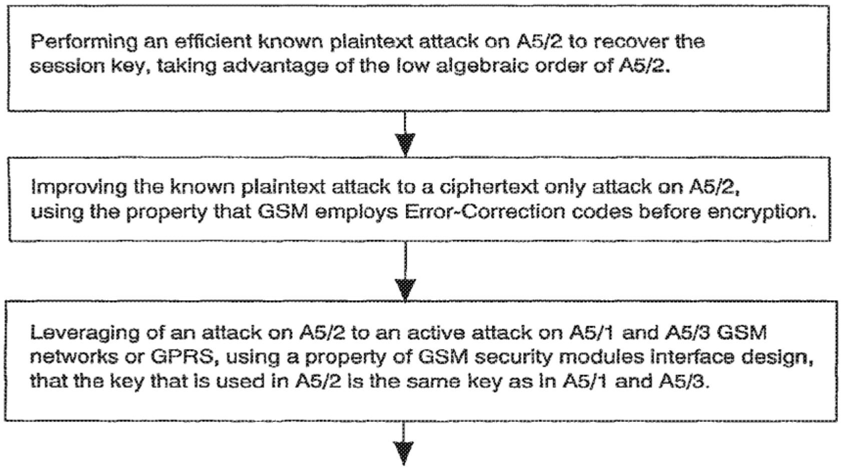
Article #15 Notes: Cryptanalysis method and system

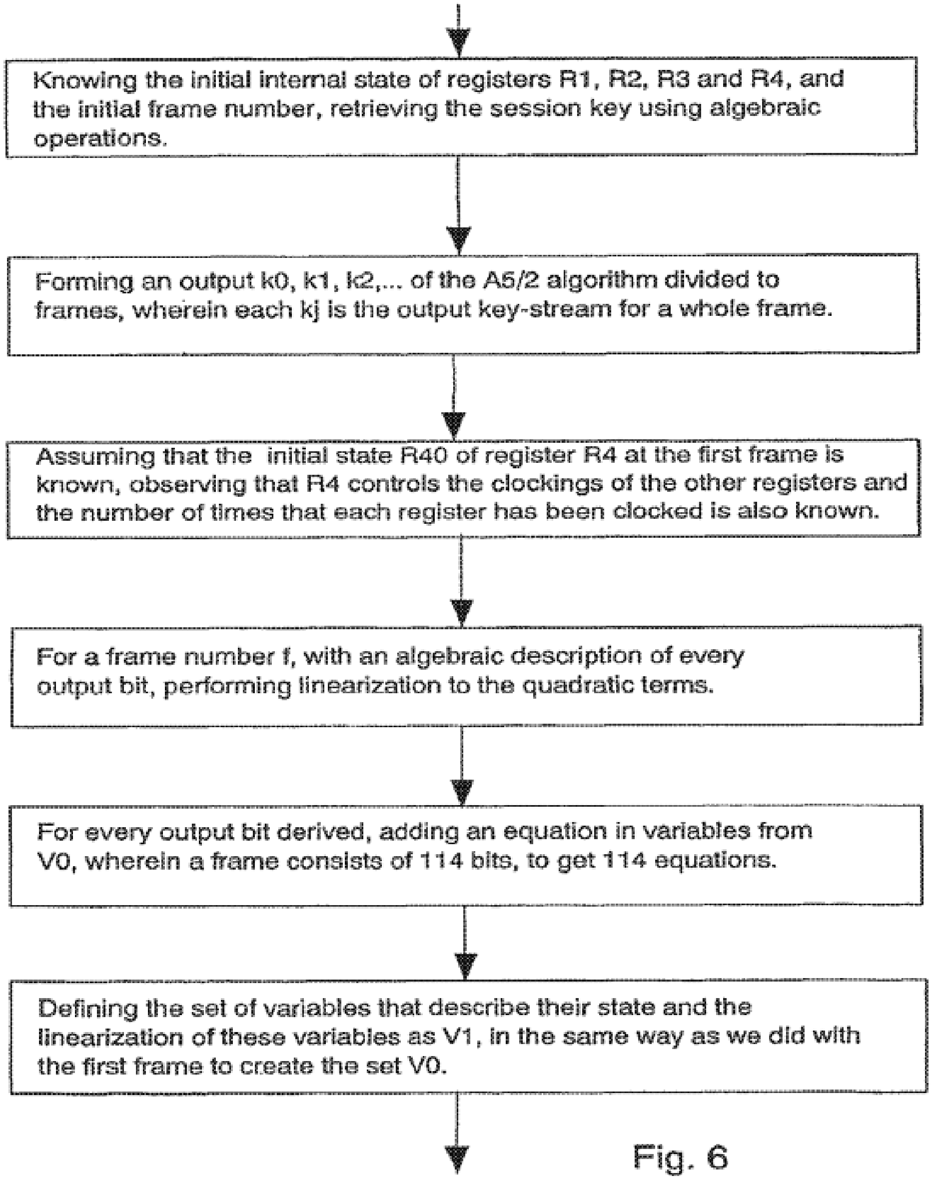
Article notes should be on separate sheets

Source Title	Cryptanalysis method and system
Source citation (APA Format)	Barkan, E., & Biham, E. (2021). *Cryptanalysis method and system* (Patent No. 10924462B2). U.S. Patent and Trademark Office. https://patents.google.com/patent/US10924462B2
Original URL	https://patents.google.com/patent/US10924462B2
Source type	Patent
Keywords	N/A
#Tags	#cs, #cryptanalysis

Summary of key points + notes (include methodology)	- GSM is a popular method of cellular communication protected by encryption by the A5 family of functions
	    - Testing the level of security of this protocol may be desirable, so an efficient cryptanalytic method is needed
	    - Stream cipher is used instead of block cipher since error correction is used, so a flipped bit will not propagate changes
	- Previous work has not been effective for all of the time; has not been able to find the session key, which is critical for extending the attack; or has been effective but not practical
	- A5/2 consists of 4 registers represented by XORed polynomials
	    - Each step of A5/2, the first 3 registers are clocked
	    - Then, the 4th register is clocked, and one output bit is ready, which is a nonlinear function of the internal states of the first three registers
	    - The values in the fourth register act as inputs for the clocking mechanism of the first 3 registers
	    - A quadratic majority function is then used to clock the first 3 registers to the 4th register
	    - 99 bits of output are discarded and 228 bits are used
	    - 114 bits are used to encrypt the connection from the network to the customer, and the 114 remaining for vice versa
	- The patent covers new methods for attacking A5/1, A5/2, and to an extent A5/3 encryption, allowing an attacker to hijack GSM communications
	- Ciphertext-only cryptanalytic attack on A5/1

	- Efficient known-plaintext attack on A5/2 - trying all possible values of the fourth register and solving the produced system of equations to deduce the values for the other three registers, which suggests the value of the key - Constructs linearized variables that describe the output of the hash through a system of equations, which is then solved using Gauss elimination - Can be optimized greatly by filtering for plausible values of the fourth register - Since GSM uses error-correction codes before encryption, the previously-described attack be adapted to a ciphertext-only attack - If the error-correction scheme is known, it can be transformed into a linear equation with the ciphertext in terms of the key - Then, substitution of each bit in the key provides the ciphertext in terms of the initial frame/block, which provides a system of equations which can then be solved by the previously described method - Since A5/1 and A5/3 use the same session key as A5/2 by design, this attack can be extended to them - Very fast in practice (milliseconds), although precomputation takes significant computing power - Can be carried out in practice with a man-in-the-middle attack - Applications include eavesdropping on or hijacking conversations - Can also be applied to GPRS (General Packet Radio Service), a newer, higher-tech service for GSM - Demonstrates the weakness of the currents algorithms used in GSM - Can possibly be patched by eliminating the error-correction code stage before encryption, which critically weakened the system - More frequent authentication may be helpful - Could use more of the available key bits for encryption
Research Question/Problem/ Need	How can known-plaintext and ciphertext-only attacks be conducted on the A5 family of functions?

Important Figures	 Performing an efficient known plaintext attack on A5/2 to recover the session key, taking advantage of the low algebraic order of A5/2. Improving the known plaintext attack to a ciphertext only attack on A5/2, using the property that GSM employs Error-Correction codes before encryption. Leveraging of an attack on A5/2 to an active attack on A5/1 and A5/3 GSM networks or GPRS, using a property of GSM security modules interface design, that the key that is used in A5/2 is the same key as in A5/1 and A5/3. Fig. 5. A method for ciphertext only attack

	 Fig. 6 Fig. 6. A Known Plaintext Attack on A5 / 2 Method
VOCAB: (w/definition)	A5 - a family of stream ciphers used in the GSM telephone standard Stream cipher - a symmetric key cipher where each digit of the plaintext is combined with a pseudo randomly-generated stream of ciphertext digits Ciphertext-only attack - cryptanalytic attack model where only a set of ciphertexts are known Frame - single communication instance
Cited references to follow up on	Ohmori, M., Matsuzaki, N., Tatebayashi, M., & Maruyama. (2002). *Block cipher using key data merged with an intermediate block generated from a*

	previous block (Patent No. 6459792B2). U.S. Patent and Trademark Office. https://patents.google.com/patent/US6459792B2/
Follow up Questions	Can this technique be applied to other stream ciphers? Is there a common approach to converting known-plaintext to ciphertext-only attacks?

Article #16 Notes: Improving Attacks on Round-Reduced Speck32/64 Using Deep Learning

Article notes should be on separate sheets

Source Title	Improving Attacks on Round-Reduced Speck32/64 Using Deep Learning
Source citation (APA Format)	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
Original URL	https://eprint.iacr.org/2019/037
Source type	Conference paper
Keywords	Deep learning, differential cryptanalysis, Speck
#Tags	#cs, #ai, #cryptanalysis, #related-work, #methods

Summary of key points + notes (include methodology)	- Calculated predicted difference distribution of Speck32/64 with specific input difference under Markov assumption for up to eight rounds, yielding good model of difference distribution of the hash
	  - Around 34 GB of distribution data for each round
	  - Validated model validity by checking highest-probability differential transition, checking true positive rates for the distinguishers on the distribution compared to an observed dataset, and by comparing distinguisher performance to those trained on 100 billion samples of empirical data
	- Produced distinguishers based on deep residual neural networks with mean key rank roughly five times lower than classical distinguishers making use of the difference distribution table produced
	  - Fairly strong distinguishers can be developed up to round 6 with very small datasets using few-shot learning - also helpful for deriving good input differences without human input
	  - Input as a matrix based on the words in the ciphertext pairs
	  - Best network architectures was a bit sliced convolution into a residual tower of two-layer convolutional neural networks, interpreted by a densely-connected prediction layer
	  - 5 and 6 rounds used depth-10 tower, while 7 and 8 rounds used

	just one residual block - Ample use of batch normalization and ReLU, with an output with sigmoid activation - Data (plaintext and keys) was generated using Linux's random number generation and encrypted - 10^7 samples - Key search used to optimize - 8-round distinguisher only slightly stronger than classical distinguisher - Distinguishers can be extended by one round using key ranking techniques and evaluated by combining their ciphertext pair scores to get key scores - A full key recovery attack on any Speck variant with a free key schedule using purely differential techniques cannot have a success rate beyond 50% -> neural distinguishers hence improve this since they are not purely differential - Developed key search scheme using Bayesian optimization that, in conjunction with the neural distinguishers, greatly reduces security of 11-round Speck - Partial key recovery attack can be conducted - Wrong key response profile (Fig. 3.) can be used for key search - Algorithm is first tried on each ciphertext structure and then iterated until the key is guessed correctly (or a limit is reached) - First round key and at most two bits off for second round key - 521 out of 1000 trials - Neural distinguishers use features of the ciphertext pair distribution unable to be detected by classical distinguishers, outside of the difference distribution table - Real differences experiment - distinguish ciphertexts given the random and real difference distributions are the same - Neural distinguishers perform better than random guessing in this experiment without any retraining, while classical distinguishers are useless - Key search technique also adapted for this situation - The neural distinguishers don't use any properties of the Speck key schedule, as they perform just as well with a free key schedule - Relatively fast training - minutes, with access to a graphics card - Transferrable to other hashes, as the networks only have knowledge of the word structure of Speck - The use of Bayesian optimization and other key search techniques are useful for any situation where the exploited function is expensive to evaluate - First paper to demonstrate a neural attack on a cipher or hash that improves upon state of the art
Research Question/Problem/ Need	Can neural-network-based distinguishers be effective on the Speck32/64 hash?

Important Figures

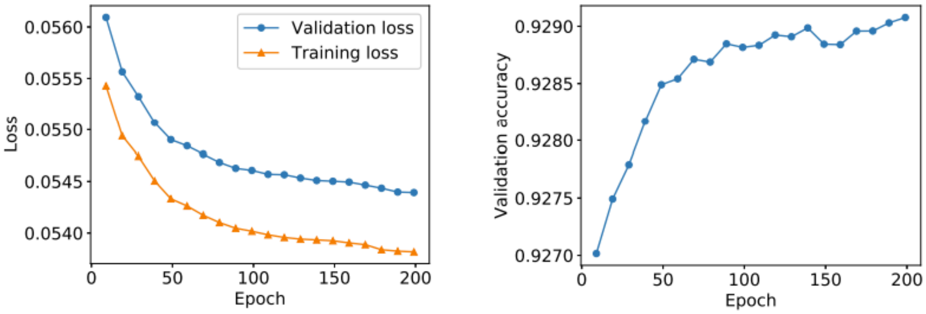


Fig. 1. Training a neural network to distinguish 5-round Speck32/64 output for the input difference Δ = 0x0040/0 from random data. (left) Training and validation loss by epoch. (right) Validation accuracy. (both) Only data for epochs with lowest learning rate is shown. Intermediate epochs contained excursions to low performance. Full learning history for this run is available from supplementary data.

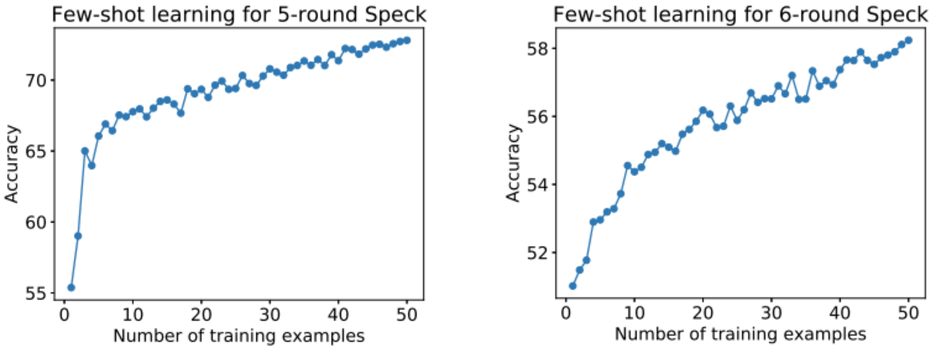
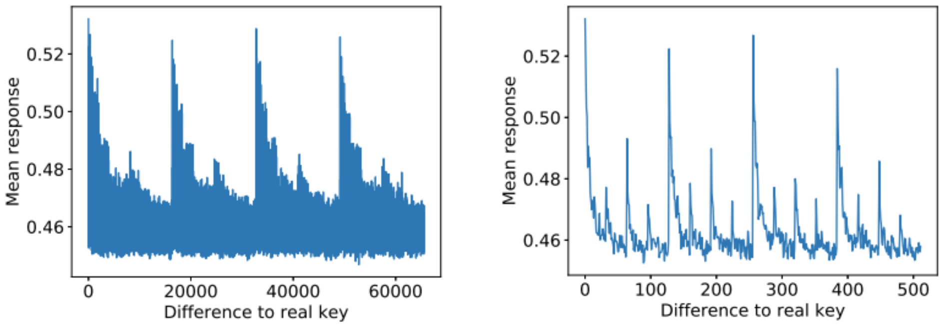


Fig. 2. Few-shot learning on the D5 and D6 tasks using a pre-trained classifier to preprocess the input data. Algorithm 2 was used with a fixed auxiliary network trained to distinguish Speck32/64 reduced to three rounds with a random fixed input difference. The number of training examples supplied was varied from 1 to 50. The accuracy figures shown are an average over 100 runs for each training set size, where for each training run a fresh training set of the indicated size was generated on the fly. Accuracy was measured against a fixed test set of size 50000. Measured accuracy is above guessing at 2σ significance level even for a single training example.

	 Fig. 3. Wrong key response profile (only μδ shown) for 8-round Speck32/64 and our 7-round neural distinguisher. For each difference δ between trial key and right key, 3000 ciphertext pairs with the input difference 0x0040/0000 were encrypted for 8 rounds of Speck using randomly generated keys and then decrypted for one round using a final subkey at difference δ to the right key. Differences are shown on the x-axis, while mean response over the 3000 pairs tried is shown in the y-axis.
VOCAB: (w/definition)	Speck - NSA block cipher family designed for IOT devices Cryptographic distinguisher - distinguishes between pairs of ciphertext and strings generated randomly Markov assumption - the assumption that the Markov property, that future evolution is independent of a system's past states, holds Residual neural network - deep neural network architecture with skip connections, designed to support hundreds or thousands of layers Bayesian optimization - design strategy for optimization of black-box functions, usually used to optimize functions that are expensive to evaluate Key schedule - subkeys in a hash used in each round
Cited references to follow up on	Greydanus, S. (2017). *Learning the Enigma with recurrent neural networks* (arXiv:1708.07576). arXiv. http://arxiv.org/abs/1708.07576 Rivest, R. L. (1993). Cryptography and machine learning. In H. Imai, R. L. Rivest, & T. Matsumoto (Eds.), *Advances in Cryptology—ASIACRYPT '91* (Vol. 739, pp. 427–439). Springer Berlin Heidelberg. https://doi.org/10.1007/3-540-57332-1_36
Follow up Questions	Can the neural distinguishers be optimized by using information from the precomputed difference distribution? Why were residual convolutional neural networks the most effective for this task?

Article #17 Notes: Double-hashing operation mode for encryption

Article notes should be on separate sheets

Source Title	Double-hashing operation mode for encryption
Source citation (APA Format)	Almuhammadi, S. A., & Amro, A. (2021). *Double-hashing operation mode for encryption* (Patent No. 10887080B2). U.S. Patent and Trademark Office. https://patents.google.com/patent/US10887080B2
Original URL	https://patents.google.com/patent/US10887080B2
Source type	Patent
Keywords	N/A
#Tags	#cs, #cybersecurity
Summary of key points + notes (include methodology)	- DHOME - Double-Hashing Operation Mode for Encryption, the invention - Big data involves extremely large and complex datasets, which require secure and performant encryption schemes - Patent documents encryption scheme for big data using double hashing, involving two different hash functions, to improve security - Supports both symmetric and asymmetric key handling - Ciphertext header structure allows efficient cloud data sharing since only the header must be re-encrypted, not the data itself - Generate a random seed of at least 512 bits, encrypt the seed with a seed key and store it in the header, hash the seed two times to acquire a key stream, and use that key stream to encrypt the data (most likely through XOR) - The first hash outputs a different number of bits than the second - The seed key could be a shared secret key or a public key, where a private key is then used to decrypt the header - The plaintext and ciphertext are split into corresponding segments with length of the output of the second hash function - The hash functions should be like those in the SHA-2 family; the MD family and the SHA-3 family may also be used - Symmetry or asymmetry can be varied based on the method the seed is encrypted with, such as AES or RSA - The second hash helps resist known-plaintext and chosen-ciphertext attacks by masking the relationship between the key stream and the

<table>
<tr>
<td></td>
<td>

values generated from the seed through the first hash

- Even if the entire key stream is revealed, the main seed and key are still secure due to the second hash

- The patent suggests SHA-384 for the first hash and SHA-512 for the second hash, based on sensitivity analysis minimum results
- AES-256 is 1.71 times slower than DHOME (a significant difference)
- Other techniques such as preimage attacks are not effective due to the double-hash structure

</td>
</tr>
<tr>
<td>Research Question/Problem/ Need</td>
<td>Can a secure encryption scheme be created by combining two different hash functions?</td>
</tr>
<tr>
<td>Important Figures</td>
<td>

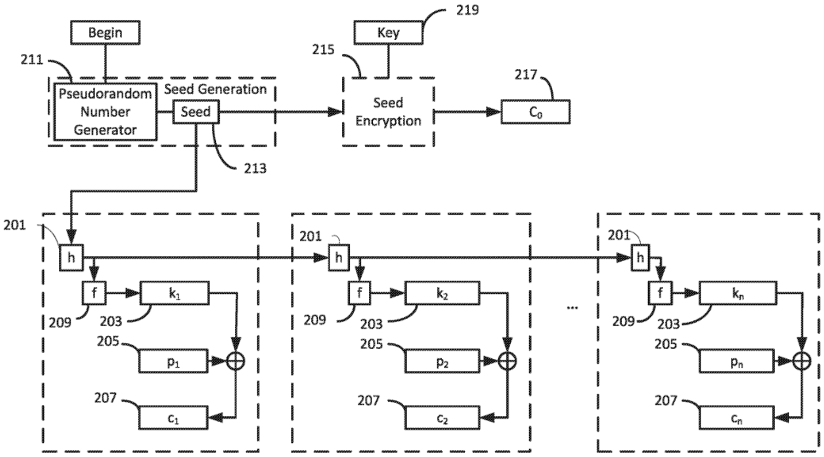


FIG. 2. Illustrates a block diagram of DHOME according to an exemplary aspect of the disclosure

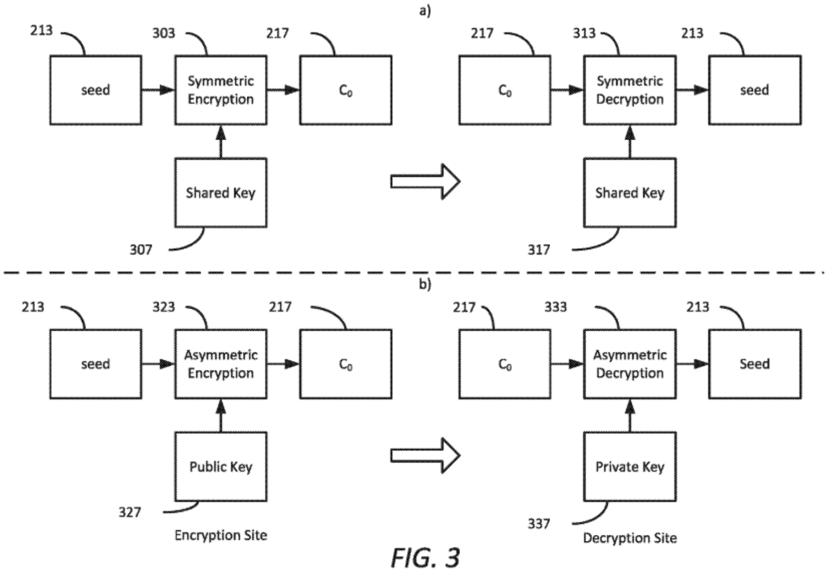


FIG. 3

FIG. 3. A flowchart illustrating seed encryption modes according to an exemplary

</td>
</tr>
</table>

<table>
<tr>
<td></td>
<td>

aspect of the disclosure

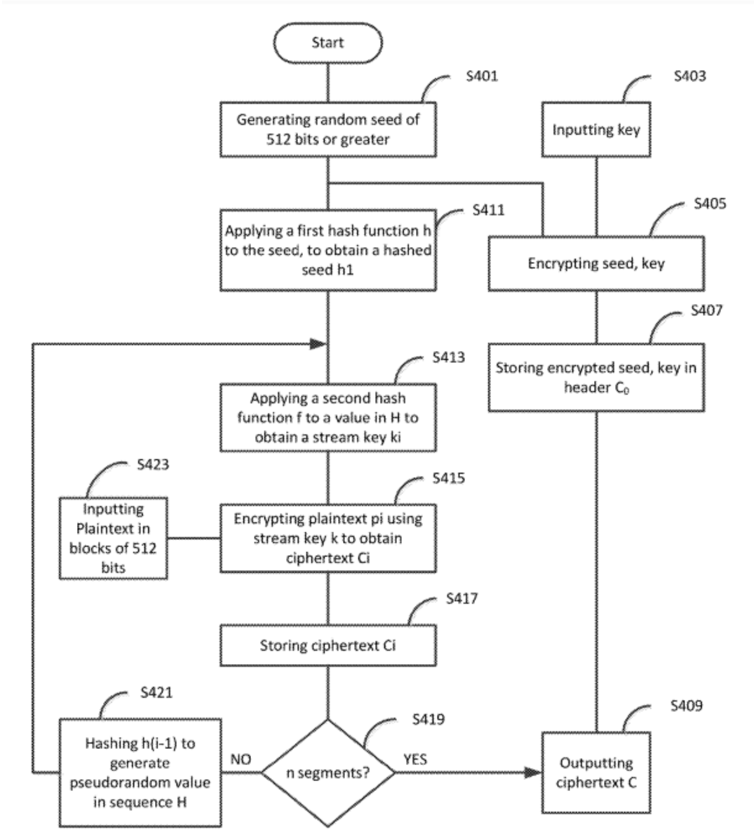


FIG. 4. A flowchart illustrating the encryption scheme according to an exemplary aspect of the disclosure

</td>
</tr>
<tr>
<td>

VOCAB: (w/definition)

</td>
<td>

Asymmetric key cipher - keys exist as public and private key pairs, where anyone can encrypt with the public key, but only those with access to the private key can decrypt
AES - Advanced Encryption Standard, a NIST block cipher succeeding DES with a size of 128 bits

</td>
</tr>
<tr>
<td>

Cited references to follow up on

</td>
<td>

Aoki, K., Guo, J., Matusiewicz, K., Sasaki, Y., & Wang, L. (2009). Preimages for step-reduced SHA-2. In M. Matsui (Ed.), *Advances in Cryptology – ASIACRYPT 2009* (Vol. 5912, pp. 578–597). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-10366-7_34

</td>
</tr>
<tr>
<td>

Follow up Questions

</td>
<td>

Can this architecture be challenged with a meet-in-the-middle attack (even if the two hash functions used are different)?
How will the seed be securely randomly generated?

</td>
</tr>
</table>

Article #18 Notes: Learning the Enigma with Recurrent Neural Networks

Article notes should be on separate sheets

Source Title	Learning the Enigma with Recurrent Neural Networks
Source citation (APA Format)	Greydanus, S. (2017). *Learning the Enigma with recurrent neural networks* (arXiv:1708.07576). arXiv. http://arxiv.org/abs/1708.07576
Original URL	https://arxiv.org/abs/1708.07576
Source type	arXiv preprint
Keywords	N/A
#Tags	#cs, #ai, #cryptanalysis, #introduction, #related-work, #methods

Summary of key points + notes (include methodology)	- The decryption process can be seen as a sequence-to-sequence translation task, so RNNs can be applied
	    - Learn algorithmic representations of complex polyalphabetic ciphers in an automated manner
	- Machine-learning-based approaches
	    - Previous work with the Vignere cipher and a simply feedforward neural network was reliant on the cipher's architecture and greatly simplified, while this paper develops a robust and generally-applicable approach
	- Train a neural network to minimize the loss between the modeled decryption function and the actual decryption method
	- Training examples were constructed by simply concatenating the key, plaintext, and ciphertext matrices
	- LSTM is sufficient to store the key in memory
	- Used a single LSTM cell with a fully-connected softmax layer
	    - Additional layers/cells were too slow
	- Loss initially decreases rapidly as the neural network learns simple statistical distributions
	    - Accuracy increases rapidly a bit later, when the model presumably begins learning the cipher itself
	    - A large portion of training is then spent on an accuracy increase of 5%
	- Three types of polyalphabetic ciphers were considered: Vignere, Autokey, Enigma
	- Training data was generated on the fly to reduce memory load and

	likelihood of overfitting
	    - Ciphertext length 14, key length 6 (3 for Enigma)
	- Used “Xavier” initialization for all hyperparameters based on previous work
	    - Mini batch stochastic gradient descent with batch size 50 and the Adam optimizer
	    - Double-checked for overfitting
	- Performs well on new phrases and messages of variable length
	- LSTM required memory size of at least 2048 units for Enigma
	- Magnitudes of hidden activations may increase linearly within the neural network, potentially contributing to lower decryption accuracy on very long sequences
	- For different keys, the Enigma hidden activations change completely, but only the magnitudes change for different messages
	    - Suggests th neural net is a switch unit that only works in certain situations
	- Examination of the hidden activations for each RNN yields that they reflect qualitative properties of their respective ciphers and are relatively unique, suggesting that the neural networks may hold interesting information about the hash it is modeling
	- The neural network for Enigma requires a significant amount of memory compared to Autokey (slightly higher since its internal representation of the key must be updated during the hash) and Vignere (lowest since it uses a static key value)
	- This work can be trivially extended to a key-recovery attack
	- Quite data inefficient - model needs at least a million training examples to learn a cipher, which is impractical
	- First general method for modeling and reversing polyalphabetic ciphers

Research Question/Problem/ Need	Can recurrent neural networks conduct cryptanalysis on polyalphabetic ciphers, most notably the Enigma cipher?

Important Figures

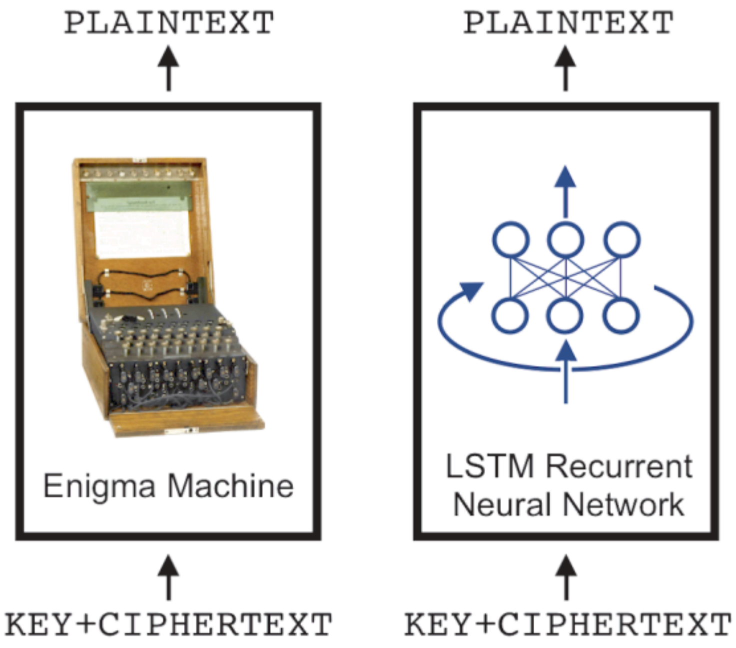
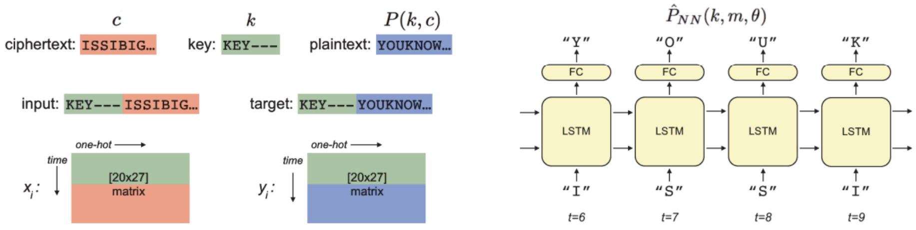


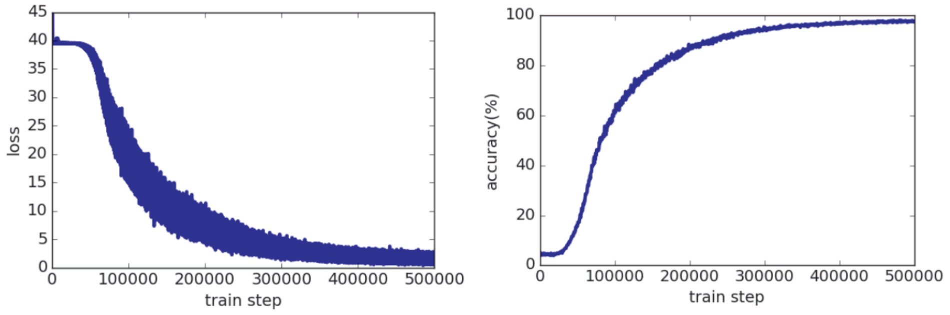
Figure 1: Our LSTM-based model can learn the decryption function of the Enigma from a series of ciphertext and plaintext examples.



(a) Data

(b) Model

Figure 2: (a) Expressing the decryption process as a sequence-to-sequence translation task. (b) Our Recurrent Neural Network (RNN) -based model unrolled for time steps 6 to 9 (FC: fully-connected layer).



(a) Train loss

(b) Test accuracy

Figure 3: Loss decreases rapidly at first, around 5000 train steps, as the network learns to capture simple statistical distributions. Later, around 100000 train steps, model learns the Enigma cipher itself and accuracy spikes. A significant portion of training, starting around 350000 train steps, is spent gaining the last 5% accuracy.

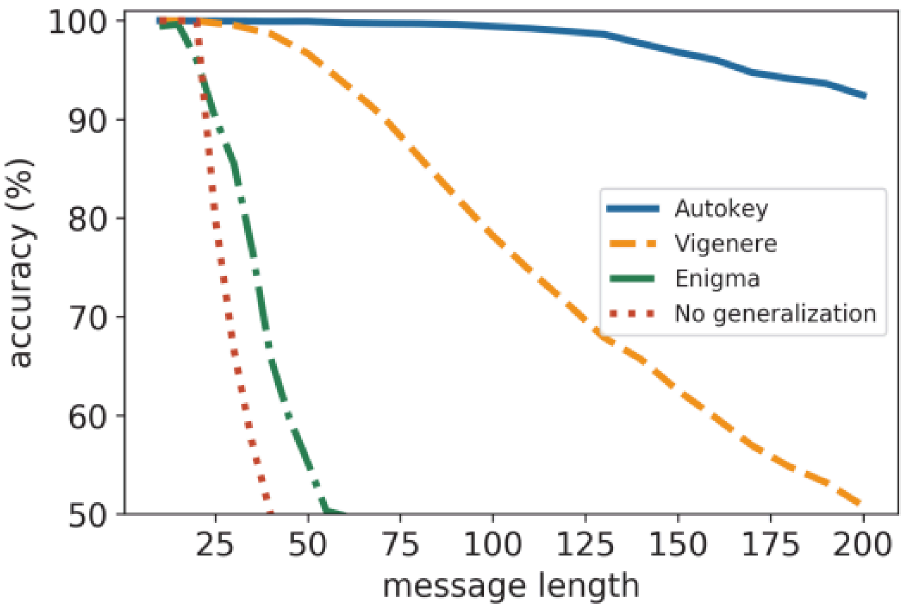
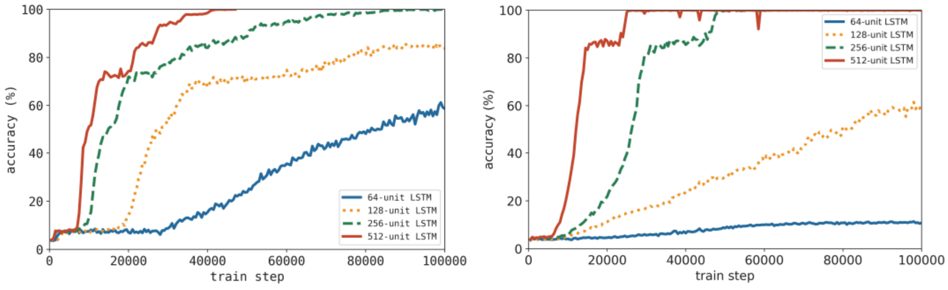


Figure 4: The model, trained on messages of 20 characters, generalizes well to messages of over 100 characters for the Vigenere and Autokey ciphers. Generalization occurs on the Enigma, but to a lesser degree, as the task is far more complex.



(a) Vigenere task

(b) Autokey task

Figure 6: Shown above are test accuracies of our model on the Vigenere and Autokey cipher tasks. Notice that for small RNN memory sizes (64 and 128 hidden units), the model achieves better performance on the Vigenere task. Meanwhile, for large memory sizes (256 and 512 hidden units), the model converges to 99+% accuracy more rapidly on the Autokey task. Evidently, the model's test accuracy is more sensitive to memory size on the Autokey task than on the Vigenere task.

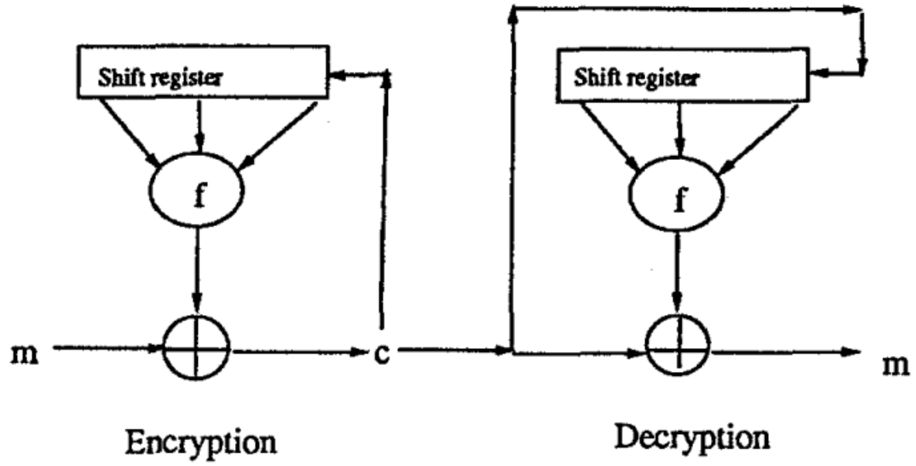
VOCAB: (w/definition)	Polyalphabetic cipher - substitution cipher using multiple alphabets Recurrent neural network - bidirectional neural network type where the outputs of some nodes affect inputs to the same nodes Long Short-Term Memory (LSTM) - RNN type based on being able to store some values in memory (easier to train in practice)
Cited references to follow up on	Kearns, M., & Valiant, L. (1994). Cryptographic limitations on learning boolean formulae and finite automata. *Journal of the ACM*, *41*(1), 67–95. https://doi.org/10.1145/174644.174647
Follow up Questions	Why are recurrent neural networks suitable for this task? Can these findings be extended past polyalphabetic ciphers, which are relatively weak compared to modern-day hash functions?

Article #19 Notes: Cryptography and machine learning

Article notes should be on separate sheets

Source Title	Cryptography and machine learning
Source citation (APA Format)	Rivest, R. L. (1993). Cryptography and machine learning. In H. Imai, R. L. Rivest, & T. Matsumoto (Eds.), *Advances in Cryptology—ASIACRYPT '91* (Vol. 739, pp. 427–439). Springer Berlin Heidelberg. https://doi.org/10.1007/3-540-57332-1_36
Original URL	https://link.springer.com/chapter/10.1007/3-540-57332-1_36
Source type	Conference paper
Keywords	N/A
#Tags	#cs, #ai, #cybersecurity, #cryptanalysis, #introduction

Summary of key points + notes (include methodology)	- Machine learning and cryptanalysis are very similar, almost like "sister" fields
	  - Cryptanalysis is essentially attempting learn an unknown function, the decryption function, given a certain amount of information, like machine learning
	  - Key and key space corresponds to target function and class of target functions, although some assumptions differ
	  - Different attack types are analogous to query types and vary in prior knowledge
	  - Exact vs approximate inference
	  - Computational complexity is extremely important - time-space tradeoffs also factor in
	  - Minimum information needed to solve the problem
	- Cryptography shows that certain classes of functions such as boolean formulas are computationally intractable
	  - For both representation-dependent and representation-independent results
	  - Representation-dependent - finding a function represented in a certain way is computationally intractable, proven NP-complete assuming P does not equal NP
	- Machine learning theory has also had an impact on cryptography
	  - Could be used to cryptanalyze simple cryptosystems
	  - Cipher-feedback systems

	- If plaintext and ciphertext pairs are known, a learning algorithm could be used to infer the function - Approximate learning is sufficient - Various learning techniques can be used to infer the function given that it belongs to a certain class, so the cryptosystem can be strengthened by avoiding these classes - Learning theory may enable more effective information compression, which strengthens cryptographic functions
Research Question/Problem/ Need	What is the connection between machine learning and the fields of cryptography and cryptanalysis?
Important Figures	 Figure 1: In cipher-feedback mode, each plaintext message bit m is encrypted by exclusive-oring it with the result of applying the function f to the last n bits of ciphertext, where n is the size of the shift register. The ciphertext bit c is transmitted over the channel; the corresponding decryption process is illustrated on the right.
VOCAB: (w/definition)	Computationally intractable - a problem that can be solved in theory but takes too many resources in practice NP - nondeterministic polynomial time, a problem that cannot be computed deterministically with polynomial time complexity Cipher-feedback system - using a block cipher as a stream cipher
Cited references to follow up on	Naor, M., & Yung, M. (1990). Public-key cryptosystems provably secure against chosen ciphertext attacks. *Proceedings of the Twenty-Second Annual ACM Symposium on Theory of Computing - STOC '90*, 427–437. https://doi.org/10.1145/100216.100273

Follow up Questions	How could modern machine-learning techniques, such as transformer architecture or denoising diffusion, be applied to cryptanalysis? If a Davies-Meyer hash uses a learnable round compression function, is the hash as a whole learnable?

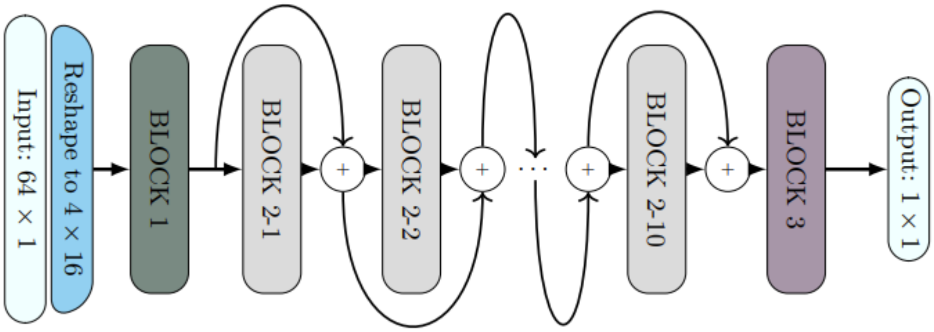
Article #20 Notes: A Deeper Look at Machine Learning-Based Cryptanalysis

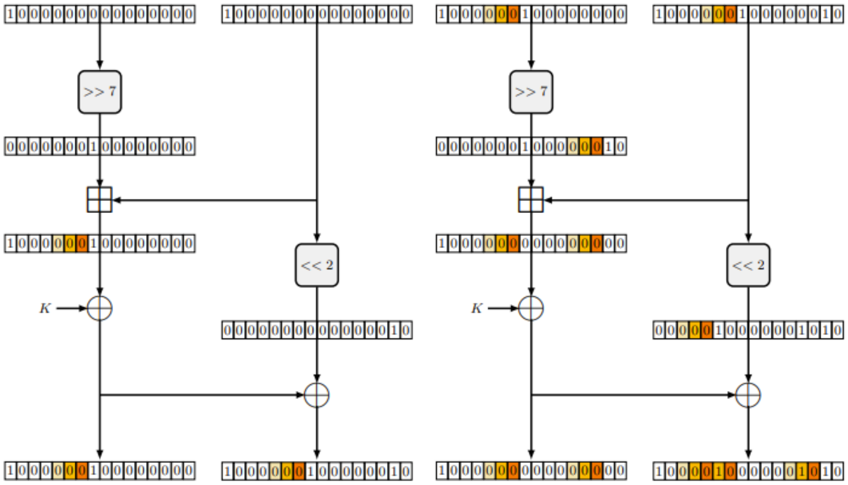
Article notes should be on separate sheets

Source Title	A Deeper Look at Machine Learning-Based Cryptanalysis
Source citation (APA Format)	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
Original URL	https://eprint.iacr.org/2021/287
Source type	Conference paper
Keywords	Differential Cryptanalysis, SPECK, Machine Learning, Deep Neural Networks, Interpretability
#Tags	#cs, #ai, #cryptanalysis, #related-work

Summary of key points + notes (include methodology)	- Gohr presented a deep neural network-based cryptographic distinguisher on the NSA block cipher Speck at CRYPTO2019 (see article #16)
	  - Improved over state-of-the-art with both the distinguisher and the corresponding key recovery attack
	  - Potential as a generic tool to carry out basic cryptanalysis of a cipher
	  - Not clear what information the neural network is actually deducing -> needs to be interpreted
	  - Unknown what extra property the neural network is using past the difference distributions
	- Specified input difference minimizes differences for 3 or 4 rounds
	- Restricted neural distinguishers to only have access to the difference distributions by changing the input to only the difference, not the pairs themselves, and found that they performed worse
	- Found that the neural distinguisher relies not only on the differential distribution of the ciphertext pairs, but also the differential distributions of the second to last and third to last rounds
	  - Analyzed ciphertext pairs and modeled difference propagation

	across rounds
	    - Retrained distinguisher with various experimental conditions
	- Non-neural Speck distinguisher using selective partial decryption for 5, 6, 7 rounds constructed based on these findings achieved same accuracy and even better efficiency compared to Gohr's findings
	    - DDT used was approximated with a dataset the same size as that used to train the neural distinguisher
	    - Trained both neural and modified distinguisher on AES-2-2-4 and obtained similar accuracies, around 60%
	- Built a simplified neural distinguisher with almost the same accuracy
	    - First tried other, easier to interpret machine learning models, with significantly lower accuracy
	    - Replaced the final MLP block with an LGBM and modified it closer to accuracy of the original - more interpretable
	    - Built new NN based on conjectured properties with same efficiency and good accuracy, but much more interpretable
	    - Performed well with the Simon block cipher as well
	- Found that the distinguisher builds a good approximation of the DDT of Speck and uses this information to classify ciphertext pairs, based on interpretability work
	- Optimized neural distinguisher by using batches of ciphertexts instead of pairs
	    - Converted input to a 2D CNN to take in more ciphertexts, significantly improving accuracy
	    - Achieved 100% accuracy for batch size 10 on round 5 and size 50 on round 6
	    - 99.7% accuracy with batch size 100 on round 7, a significant improvement over the original
	    - Used same size dataset/amount of ciphertexts as original
	- The neural distinguisher is not using a novel cryptanalytic technique, but optimizing information extraction

Research Question/Problem/ Need	How does the Speck32/64 neural distinguisher work internally, and how can it be simplified and improved?
Important Figures	 Fig. 2: The whole pipeline of Gohr's deep neural network. Block 1 refers to the

	initial convolution block, Block 2-1 to 2-10 refer to the residual block and Block 3 refers to the classification block.  Fig. 7: The left (resp. right) part shows how the active bit from difference 0x8000/8000 (resp. 0x8100/8102) propagates to difference 0x8100/8102 (0x8000/820a). The darker the color, the higher the probability (≥ 1/4) that it has a carry propagated to.
VOCAB: (w/definition)	Differential Distribution Table - table describing frequencies of output differences based on generated plaintext pairs with a certain input difference Selective partial decryption - partial decryption is performed using a hypothesis on the subkey of the last round and filtered using a precomputed DDT Light Gradient Boosting Machine - gradient-boosting framework based on decision tree algorithms Gradient-boosting - ML technique using an ensemble of weak prediction models to form strong predictions
Cited references to follow up on	Maghrebi, H., Portigliatti, T., & Prouff, E. (2016). Breaking cryptographic implementations using deep learning techniques. In C. Carlet, M. A. Hasan, & V. Saraswat (Eds.), *Security, Privacy, and Applied Cryptography Engineering* (Vol. 10076, pp. 3–26). Springer International Publishing. https://doi.org/10.1007/978-3-319-49445-6_1
Follow up Questions	Will neuro-cryptanalysis always model existing approaches? If CNNs, which are traditionally used for computer vision, can be of use in cryptanalysis, how can other complex architectures, such as transformers, potentially be used?

Article #21 Notes: Does machine learning need fuzzy logic?

Article notes should be on separate sheets

Source Title	Does machine learning need fuzzy logic?
Source citation (APA Format)	Hüllermeier, E. (2015). Does machine learning need fuzzy logic? *Fuzzy Sets and Systems*, *281*, 292–299. https://doi.org/10.1016/j.fss.2015.09.001
Original URL	https://www.sciencedirect.com/science/article/pii/S0165011415004133
Source type	Journal paper
Keywords	Fuzzy sets, fuzzy logic, machine learning
#Tags	#cs, #fuzzbits, #introduction

Summary of key points + notes (include methodology)	Note: although slightly outside of my field of interest, I am investigating this area due to the suggestion by Goncharov, 2019 to improve the effectiveness of neural networks performing cryptanalysis by employing fuzzy logic
	- Fuzzy machine learning - emerged after the advent of fuzzy logic as a field
	  - Shift from knowledge-based manual design to data-driven automatic construction
	  - Fuzzy logic usually used for deductive reasoning, while machine learning is inductive
	- Majority of fuzzy ML papers are about fuzzification of standard methods such as decision trees or nearest neighbor estimation
	  - The increased flexibility may improve accuracy
	  - Less ties for decision trees, which is useful in ranking
	  - Fuzzification is usually straightforward
	  - Fuzzy models still implement standard functions, mapping normal input to normal output, so their benefits are not very apparent
	  - May cause an increase in computational complexity and risk of overfitting
	  - Sometimes shallow link to actual fuzzy logic, excluding fuzzy rule induction and decision tree learning
	- Interpretability is a core argument for fuzzy ML
	  - Fuzzy models, which are rule-based, often have far too many rules and complicated weighting/aggregation schemes, making them not as interpretable as they first appear
	  - Fuzzy sets are strongly influenced by their dataset and do not

	often produce semantically meaningful clusters to interpret
	  - Interpretability cannot be translated from knowledge-based to data-driven models
	- Epistemic uncertainty is incomplete knowledge about the true relationship between the input and output, which can be modeled with fuzzy sets of candidate models
	  - However, current research on this fails to justify how exactly it addresses uncertainty and is difficult to test empirically
	- Learning from fuzzy data is again not well-defined, although it may be a useful tool
	  - Such data might not be readily available, although there are some interesting adaptation methods
	  - A grayscale image, for example, can be interpreted as a fuzzy set
	  - Precise data can be fuzzified to control the influence of individual observations
	- Fuzzy modeling can be applied to much more than the current literature, such as modeling the data space in terms of fuzzy logic for learning algorithms to then operate on more efficiently
	- Fuzzy logic may be suitable for non-inductive learning such as transfer learning, as it can represent the knowledge process transfer
	- Fuzzy theories on uncertainty, such as imprecise probability, may be able to complement current probabilistic methods in machine learning
	  - Especially applicable to my project since my function is difficult to model and the model distribution may not be fully known
	- Although fuzzy machine learning still needs to be formalized and further scrutinized, it has the potential for interesting contributions to machine learning

Research Question/Problem/ Need	Does machine learning need fuzzy logic?
Important Figures	N/A
VOCAB: (w/definition)	Fuzzy machine learning - fuzzy systems used in ML Fuzzy set - sets where elements have varying degrees of membership Imprecise probability - generalization of probability theory to allow for partial specifications
Cited references to follow up on	Berlanga, F. J., Rivera, A. J., Del Jesus, M. J., & Herrera, F. (2010). GP-COACH: Genetic Programming-based learning of COmpact and ACcurate fuzzy rule-based classification systems for High-dimensional problems. *Information Sciences*, *180*(8), 1183–1200. https://doi.org/10.1016/j.ins.2009.12.020

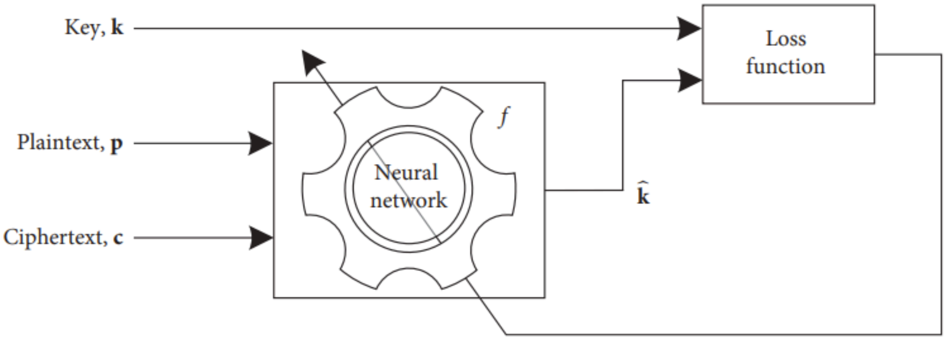
	Hüllermeier, E. (2014). Learning from imprecise and fuzzy observations: Data disambiguation through generalized loss minimization. *International Journal of Approximate Reasoning*, *55*(7), 1519–1534. https://doi.org/10.1016/j.ijar.2013.09.003
Follow up Questions	Can the fuzzy conception of uncertainty be applied to approximate some parts of complex, high-dimensional distributions given insufficient data? Does fuzzification aid in gradient descent by providing granularity? In what way is a fuzzy distribution optimized—what is the definition of optimization for such a distribution?

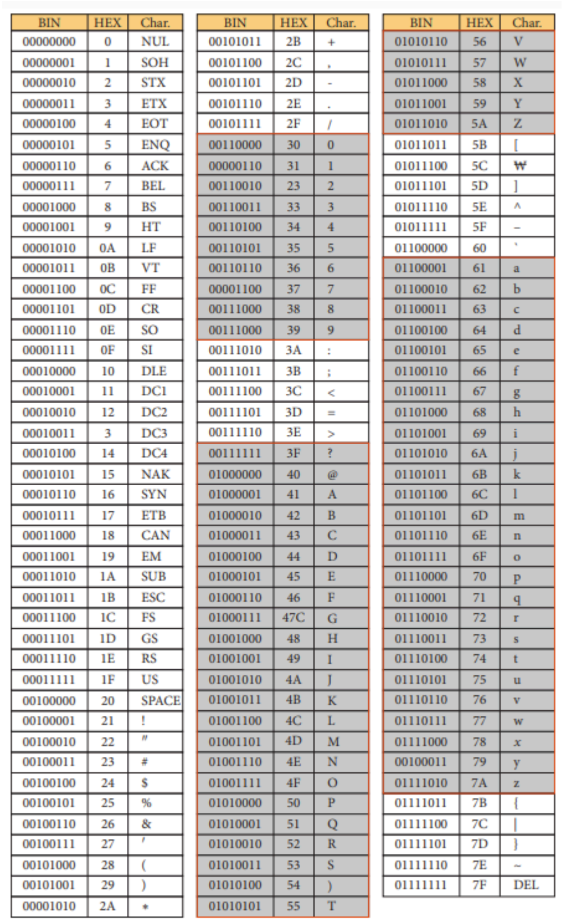
Article #22 Notes: Deep Learning-Based Cryptanalysis of Lightweight Block Ciphers

Article notes should be on separate sheets

Source Title	Deep Learning-Based Cryptanalysis of Lightweight Block Ciphers
Source citation (APA Format)	So, J. (2020). Deep learning-based cryptanalysis of lightweight block ciphers. *Security and Communication Networks*, *2020*, 1–11. https://doi.org/10.1155/2020/3701067
Original URL	https://www.hindawi.com/journals/scn/2020/3701067/
Source type	Journal paper
Keywords	N/A
#Tags	#cs, #ai, #cryptanalysis, #related-work, #methods

Summary of key points + notes (include methodology)	- Deep learning model attempts to find key based on plaintext-ciphertext pairs of simplified DES, Simon, and Speck block ciphers
	- Keyspace is restricted to 64 ASCII characters, or 512 bits
	- First model successfully to break full rounds of Simon32/64 and Speck32/64
	  - Traditional cryptanalysis does not have keyspace restriction or a text-based key
	- Impractical to generalize or automate classical cryptanalysis, but being able to quickly check the security of lightweight block ciphers for IOT is critical
	- Standard DNN with ReLU activation
	  - Inputs layer neurons match to bits of the plaintext and ciphertext
	  - Output layer neurons match to bits of the key
	  - Estimated key is hence an iterated nonlinear transformation of the input data
	  - MSE loss function - minimize difference between output and true key
	  - Data is generated from publicly-available algorithms - testing data is ciphertext plaintext pairs generated from different keys
	    - Plaintext is a random binary sequence
	    - Keys are textual and not all ASCII characters are used (see Figure 4), meaning the probability that a certain bit is 1 isn’t simply 0.5

	- BAP used to evaluate - Manually specified hyperparameters determined through testing - 5 hidden layers of 512 neurons each, with 5000 epochs - Adam optimization - Given M known plaintexts, the key is found through majority decision with the DNN - S-DES - 50k training, 10k testing samples - Evaluated with both textual key and random key - textual key was successful - Simon and Speck32/64 - 5*10^5 training, 10^6 testing samples - Attack with random key fails - Attack with textual key is successful - Reduced search space greatly, but with large drawback of text-base key, which is uncommon in practice - Restricted keyspace necessary given complexity of ciphers
Research Question/Problem/ Need	Can a generic known-plaintext attack be developed using deep learning on lightweight block ciphers?
Important Figures	 Figure 1: A schematic diagram of the DL-based cryptanalysis.



BIN	HEX	Char.	BIN	HEX	Char.	BIN	HEX	Char.
00000000	0	NUL	00101011	2B	+	01010110	56	V
00000001	1	SOH	00101100	2C	,	01010111	57	W
00000010	2	STX	00101101	2D	-	01011000	58	X
00000011	3	ETX	00101110	2E	.	01011001	59	Y
00000100	4	EOT	00101111	2F	/	01011010	5A	Z
00000101	5	ENQ	00110000	30	0	01011011	5B	[
00000110	6	ACK	00000110	31	1	01011100	5C	₩
00000111	7	BEL	00110010	23	2	01011101	5D	]
00001000	8	BS	00110011	33	3	01011110	5E	^
00001001	9	HT	00110100	34	4	01011111	5F	–
00001010	0A	LF	00110101	35	5	01100000	60	`
00001011	0B	VT	00110110	36	6	01100001	61	a
00001100	0C	FF	00001100	37	7	01100010	62	b
00001101	0D	CR	00111000	38	8	01100011	63	c
00001110	0E	SO	00111000	39	9	01100100	64	d
00001111	0F	SI	00111010	3A	:	01100101	65	e
00010000	10	DLE	00111011	3B	;	01100110	66	f
00010001	11	DC1	00111100	3C	<	01100111	67	g
00010010	12	DC2	00111101	3D	=	01101000	68	h
00010011	3	DC3	00111110	3E	>	01101001	69	i
00010100	14	DC4	00111111	3F	?	01101010	6A	j
00010101	15	NAK	01000000	40	@	01101011	6B	k
00010110	16	SYN	01000001	41	A	01101100	6C	l
00010111	17	ETB	01000010	42	B	01101101	6D	m
00011000	18	CAN	01000011	43	C	01101110	6E	n
00011001	19	EM	01000100	44	D	01101111	6F	o
00011010	1A	SUB	01000101	45	E	01110000	70	p
00011011	1B	ESC	01000110	46	F	01110001	71	q
00011100	1C	FS	01000111	47C	G	01110010	72	r
00011101	1D	GS	01001000	48	H	01110011	73	s
00011110	1E	RS	01001001	49	I	01110100	74	t
00011111	1F	US	01001010	4A	J	01110101	75	u
00100000	20	SPACE	01001011	4B	K	01110110	76	v
00100001	21	!	01001100	4C	L	01110111	77	w
00100010	22	"	01001101	4D	M	01111000	78	x
00100011	23	#	01001110	4E	N	00100011	79	y
00100100	24	$	01001111	4F	O	01111010	7A	z
00100101	25	%	01010000	50	P	01111011	7B	{
00100110	26	&	01010001	51	Q	01111100	7C	|
00100111	27	'	01010010	52	R	01111101	7D	}
00101000	28	(	01010011	53	S	01111110	7E	~
00101001	29	)	01010100	54	)	01111111	7F	DEL
00001010	2A	*	01010101	55	T			

Figure 4: Characters used in the text key generation.

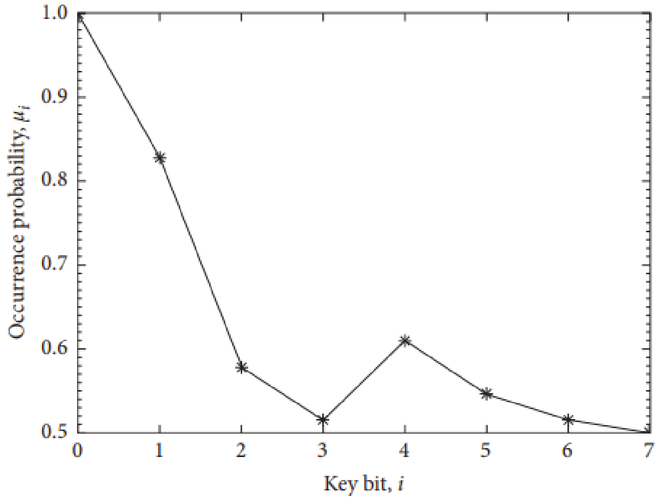


Figure 5: Occurrence probability in the text key generation.

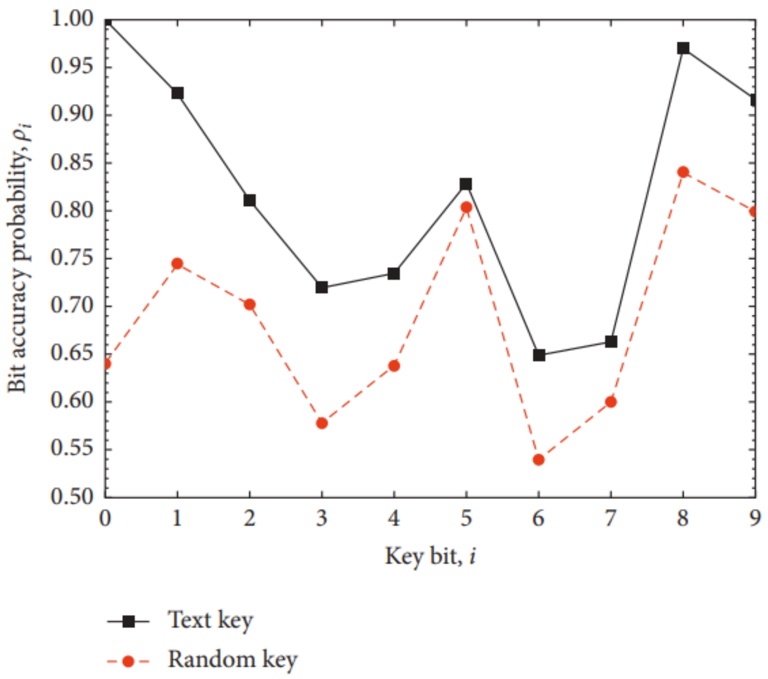
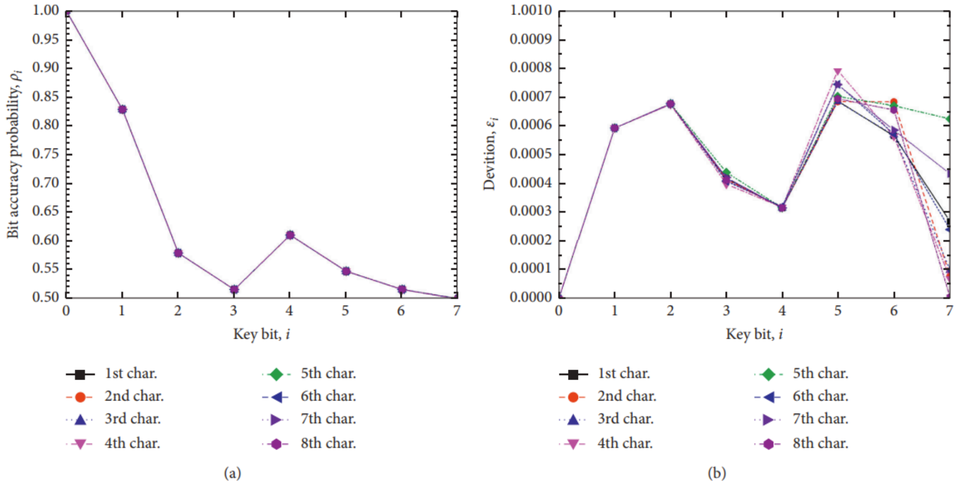


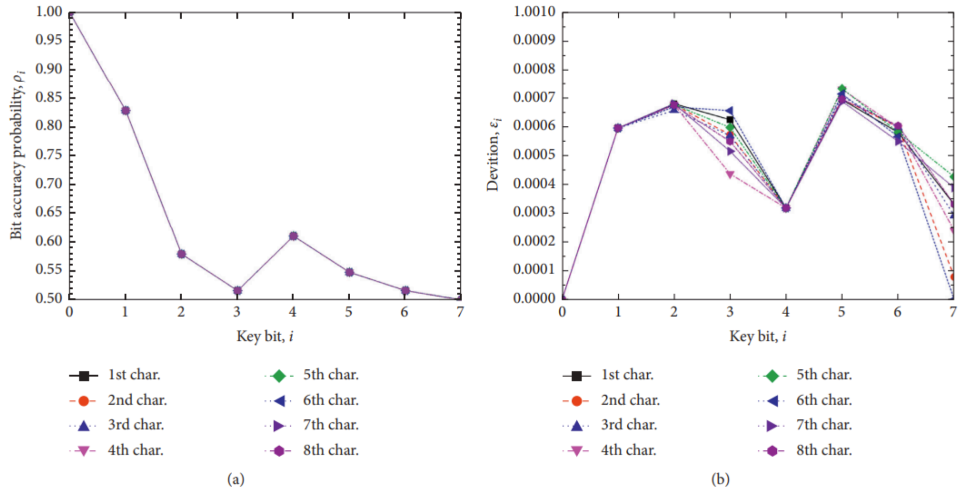
Figure 7: Bit accuracy in the S-DES with a random key.



(a)

(b)

Figure 10: Bit accuracy probability and deviation of the Simon32/64 with a text key.

	 Figure 12: Bit accuracy probability and deviation of the Speck32/64 with a text key
VOCAB: (w/definition)	Bit accuracy probability - the number of testing samples where a certain bit in the output is correct, over the total number of testing samples
Cited references to follow up on	Bafghi, A. G., Safabakhsh, R., & Sadeghiyan, B. (2008). Finding the differential characteristics of block ciphers with neural networks. *Information Sciences*, *178*(15), 3118–3132. https://doi.org/10.1016/j.ins.2008.02.016 Gomez, A. N., Huang, S., Zhang, I., Li, B. M., Osama, M., & Kaiser, L. (2018). *Unsupervised cipher cracking using discrete GANs* (arXiv:1801.04883). arXiv. http://arxiv.org/abs/1801.04883 Hu, X., & Zhao, Y. (2018). Research on plaintext restoration of AES based on neural network. *Security and Communication Networks*, *2018*, 1–9. https://doi.org/10.1155/2018/6868506
Follow up Questions	Are the models trained on textual keys at all effective for random keys? Can an effective model be trained based on a known, skewed key distribution, and does this have applications?

Article #23 Notes: Applications of SAT Solvers in Cryptanalysis: Finding Weak Keys and Preimages

Article notes should be on separate sheets

Source Title	Applications of SAT Solvers in Cryptanalysis: Finding Weak Keys and Preimages
Source citation (APA Format)	Lafitte, F., Nakahara, J., & Van Heule, D. (2014). Applications of SAT solvers in cryptanalysis: Finding weak keys and preimages. *Journal on Satisfiability, Boolean Modeling and Computation*, *9*(1), 1–25. https://doi.org/10.3233/SAT190099
Original URL	https://content.iospress.com/articles/journal-on-satisfiability-boolean-modeling-and-computation/sat190099
Source type	Journal paper
Keywords	SAT solvers, weak keys, preimage attacks, automated cryptanalysis, algebraic cryptanalysis
#Tags	#cs, #cryptanalysis, #preimage-attack, #introduction

Summary of key points + notes (include methodology)	- Relevant to my project as another way to conduct a preimage attack, as I'm currently exploring ways to extend my method
	- Introduce an efficient, generic, automated method for representing cryptographic computations as SAT problems
	  - Open sourced in the package cryptosat
	  - The relationship between the input and output bits can be expressed as an SAT problem and the key bits can be solved for
	  - Generated using C++ operator overloading to encode the problem
	  - Applied SAT solver cryptominsat to 250 random message blocks encrypted by MD4
	- Applied SAT solvers to finding weak keys in block ciphers (solving a previously open problem) and conducting a preimage attack against hashes
	- Allows discovery of weak key classes or proof they don't exist under both differential and linear attacks of full-round block ciphers WIDEA-n and MESH-64(8)
	  - Weak key causes some subkeys to have value 0 or 1, turning multiplication into a linear operation
	  - Results in a collision attack with only two compression function

    computations
    - Issue in design of function because weakness propagates from key to ciphertext
- Found several classes of weak keys in full-round WIDEA-4
    - Weak subkeys must be 0 for their first 15 most significant bits
    - Found requirements for weak key by requiring 18 weak subkeys and then solving with SAT solver
    - Thousands of weak keys were found, taking just a few seconds each, contradicting the claim that no weak keys exist for WIDEA
    - Each weak key corresponds to a different, non-overlapping class
        - WIDEA-4 is composed of 4 instances of the IDEA cipher, and each key can only be used to attack one of these instances
        - Trying to solve for all instances at once yields an unsatisfiable problem, so no such weak keys exist
        - No weak keys are expected to exist for two instances at once as well
    - Weak keys were similarly found in WIDEA-8
- Proved that a certain class of weak keys don't exist in MESH-64(8)
    - Same scheme as WIDEA - weak keys follow the 1-round pattern of differences/linear bit masks (depending on differential vs linear angle) being untransformed with maximum probability
    - SAT was unsatisfiable, so that particular class of weak keys does not exist for MESH-64(8)
- Conducted preimage attack on reduced MD4
    - Could invert up to 31 rounds in a few hours using a personal computer

Research Question/Problem/ Need

Can SAT solvers be used in cryptanalysis to identify weak keys and conduct preimage attacks?

Important Figures

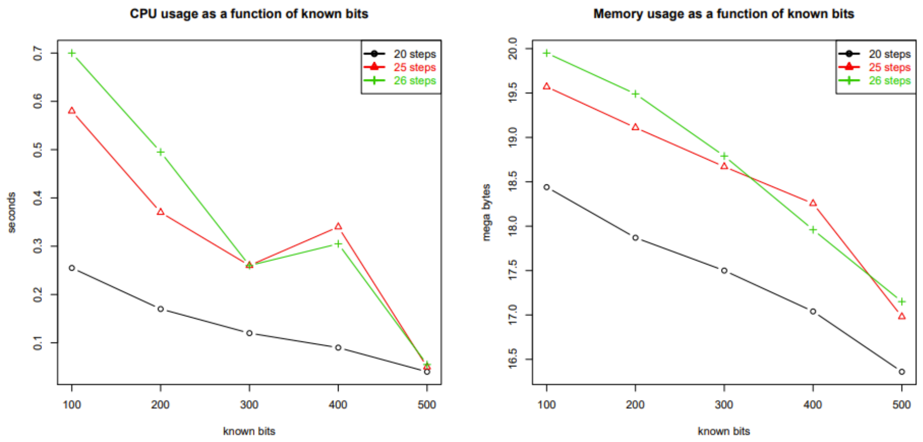
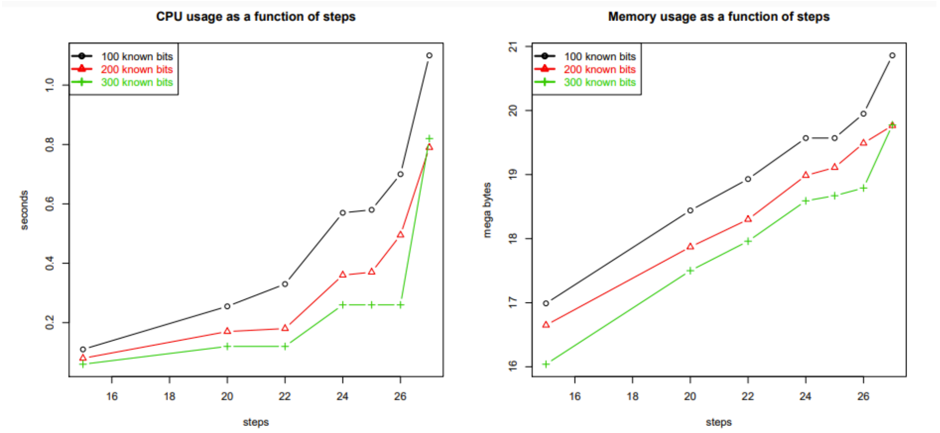


Figure 2. The median CPU time (seconds) and memory (MB) used by the SAT solver over 250 random instances, as a function of the number of unknown bits b. Each

	curve corresponds to a different value for parameter s.  Figure 3. The median CPU time (seconds) and memory (MB) used by the SAT solver over 250 random instances, as a function of the number of steps s. Each curve corresponds to a different choice of parameter b.
VOCAB: (w/definition)	SAT - boolean satisfiability problem - determining if a boolean formula can be filled in in a way that evaluates to true (similar to solving an algebraic equation, but with boolean values) Weak key - a key that leads to nonrandom cipher behavior
Cited references to follow up on	Mironov, I., & Zhang, L. (2006). Applications of SAT solvers to cryptanalysis of hash functions. In A. Biere & C. P. Gomes (Eds.), *Theory and Applications of Satisfiability Testing—SAT 2006* (Vol. 4121, pp. 102–115). Springer Berlin Heidelberg. https://doi.org/10.1007/11814948_13 Soos, M., Nohl, K., & Castelluccia, C. (2009). Extending SAT solvers to cryptographic problems. In O. Kullmann (Ed.), *Theory and Applications of Satisfiability Testing—SAT 2009* (Vol. 5584, pp. 244–257). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-02777-2_24
Follow up Questions	How do SAT solvers work internally? Can machine learning be applied to optimize them? How can the weak key search process be generalized and automated to search for all classes of weak keys?