Using AI for Block Cipher Cryptanalysis

(Improving Attacks on Round-Reduced Speck32/64 Using Deep Learning, CRYPTO 2019)

Dr. Aron Gohr
BSI

November 1st, 2019

Structure of This Talk

Highlights

- We use machine learning for **differential cryptanalysis** on Speck32/64. This yields real-or-random distinguishers that **exceed very strong classical baselines on a well-studied primitive**.
- We develop an extremely selective key search policy that uses the ML distinguishers effectively. This leads to a key recovery attack that uses very few trial decryptions (**millions of times less** than previous state of the art).
- The resulting attack on Speck32/64 reduced to 11 rounds is ≈ 200 **times faster** than previous records.
- **Manual cryptanalysis** is used to define learning tasks, design the overall attack, and extend trained distinguishers to more rounds.
- **Supplementary code and data** available on github: `https://www.github.com/agohr/deep_speck/`

Machine Learning in Cryptology I

"Neural nets work well in structured environments where there is something to learn, but not in the high-entropy, seemingly random world of cryptography."

— Schneier, *Applied Cryptography*, 20th Anniversary Edition, 2015

"Neural networks are generally not meant to be great at cryptography. "

— Abadi and Andersen, *Learning to Protect Communications With Adversarial Neural Cryptography*, arxiv 2016, `https://arxiv.org/pdf/1610.06918.pdf`

- However, many works on side-channel analysis.

Machine Learning in Cryptology II

- Rivest (Asiacrypt 1991) gave an invited paper generally on connections between ML and cryptography.
- Klimov, Mityagin and Shamir (Asiacrypt 2002) used neural networks to break a public-key encryption scheme that is itself based on neural networks.
- Greydanus (2017) trained a recurrent neural network to simulate an Enigma machine with most settings of the Enigma fixed.
- Gomez et al. showed that GANs can break Vigenere ciphers in an unsupervised setting (ICLR 2018).

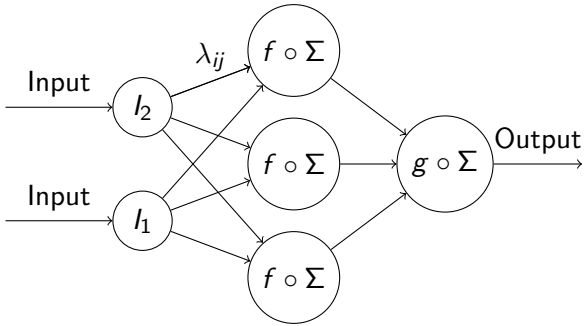
Machine Learning and AI

- Machine learning aims at making agents *learn from experience*
- AI: use heuristic methods (sometimes derived by machine learning) to solve problems that humans solve by thinking. Regularly involves online search over large solution spaces.
- Very useful for *some* problems! Examples: speech recognition, car driving, image captioning, automatic translation...
- However, not difficult to find *simple* problems where *naive* ML approaches fail or struggle badly to find a solution (e.g. calculate parity of 64-bit number).
- Spectacular successes (e.g. go, poker, translation,...) mostly use machine learning as one (crucial) *part* of a larger problem-solving system.

Machine Learning

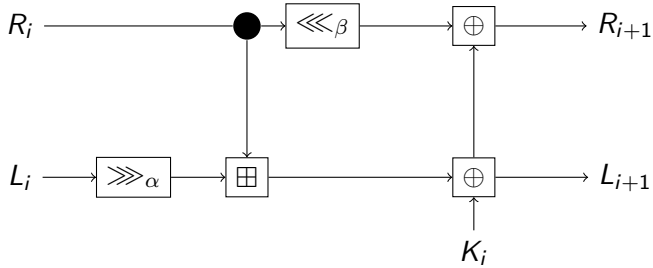
- Given some training data, search a large hypothesis space to find a model that fits the training data
- Test your model on *other* data.
- Many methods: linear regression, decision trees, random forests, support vector machines, neural networks...
- Different settings: supervised/unsupervised learning, reinforcement learning, adversarial learning...
- No single dominant strategy that fits all problems; instead, different methods with different strengths and weaknesses (although AutoML is getting better and can sometimes achieve impressive results).

Neural Networks



- Efficiently differentiable function approximators
- Search for internal weights minimizing *loss* on *training data* by applying (variants of) stochastic gradient descent.

Speck32/64



- Lightweight ARX construction, 22 rounds, $\alpha = 7, \beta = 2$, 16-bit words, 64-bit key
- Nonlinear key schedule that reuses the round function

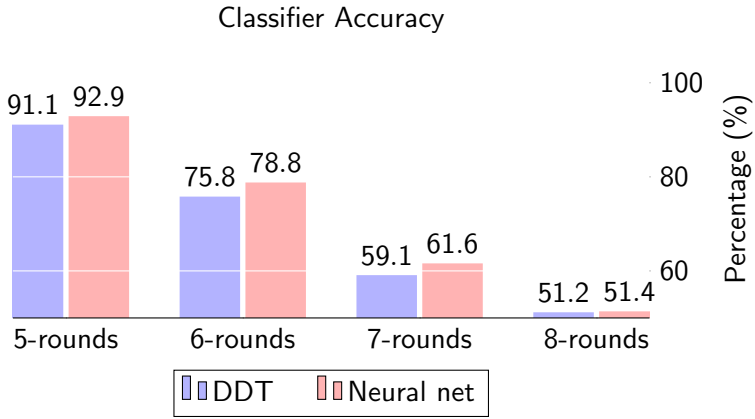
Attacks on Speck32/64: Prior Work

- 11 rounds: $\approx 2^{46}$ reduced Speck encryptions time cost on a PC using 2^{14} chosen plaintexts on average (Dinur, SAC 2014). Expected time for 12,13,14 rounds given as $2^{51}, 2^{57}, 2^{63}$ in the same model. Attacks depend on the key schedule being efficiently invertible.
- Ours: breaks 11 rounds in 15 minutes on average on one thread of one desktop CPU, or about 2^{38} reduced Speck encryptions, with very similar data usage.
- Our attack does not use the key schedule, i.e. it works with essentially the same complexity also against the free key schedule.

Training a Distinguisher for Reduced Speck

- Generate a few million real and random samples with the input difference $\Delta = (0x40, 0x0)$ (this takes a few seconds on any reasonable computer).
- Train a *deep residual neural network* to distinguish the real from the random samples.
- For 5 to 7 rounds of Speck encryption, better classifier than DDT after a few minutes of training on GPU (GTX 1080 Ti).
- For seven rounds, a more expensive training scheme gives significant further improvement to network performance.
- Fails for 8 rounds. Curricular training saves the day.

Results



Turning a Distinguisher Into a Tool for Key Recovery

Suppose we can tell whether some ciphertext C is random or has been encrypted from a known or chosen plaintext by n rounds of some block cipher. What can we do with this?

- Basic idea: attack $n + 1$ rounds by taking back the last round (trying all possible last-round keys).
- Maybe in the process, we can avoid trying *all* last round keys for more efficiency.
- Maybe we can gain some rounds by manipulating the inputs of the cipher.
- We do both in our attack.

9-round Attack: Results

Table: Statistics on a key recovery attack on 9-round Speck32/64.

	Mean	Median	SR
DDT	263.9 ± 77.7	9.0	0.13
NN	52.1 ± 34.7	1.0	0.358

- 64 chosen plain text pairs, 1000 trials for each distinguisher
- Mean key rank five times lower for NN, but high variability for both distinguishers
- Success rate (SR) is rate of key rank zero.

Deriving a key search policy

- In modern cryptography, it is often assumed that decryption under a *wrong* key gives no information on the right key.
- However, if we take back only a single round *and* use a distinguisher that extracts a little bit of information from each partially decrypted text instead of waiting for some very unlikely special event, things may be different.
- Key search policy efficiently leverages information gleaned from wrong-key responses.
- Basic idea: use strength of distinguisher response to get some information on likely errors in the key. Derive new keys to test, test them, and repeat (a small number of times).

Building an 11-round Attack

- Extend 9-round attack to 11 rounds by adding a two-round initial differential trail.
- Recreate conditions of 9-round attack by using neutral bits for the initial two-round trail.
- Apply key search policy to derive a shortlist of key candidates.
- Detect success by looking at distinguisher scores returned. If scores look good, derive shortlist of key candidates for one more round.
- If scores of some second-round candidate sufficiently high, output key guess. Otherwise, ask for more data (or look again at data already acquired).

Open Questions

- Can we automatically learn known-plaintext distinguishers?
- Which other machine learning methods will work on these problems?
- How much knowledge can be extracted from the results of machine learning in this context?
- To what extent can we use AI techniques to improve key search strategies on cryptographic problems?
- Classical ciphers are commonly attacked by combining specialised statistical tests with manual partitioning of the key space and generic optimization algorithms (e.g. hill-climbing, simulated annealing). Can we improve on attacks on classical ciphers by using optimization techniques that leverage precomputation about the cipher?

Conclusions

- Machine learning worked really well in this instance.
- NN efficiently exploits ciphertext pair distribution.
- Choosing the right learning task and choosing a good model structure crucial for success. Manual cryptanalysis crucial for deriving competitive attack from results.
- Automatically derived efficient key search policy.
- Neural networks are not *black* boxes (e.g. few-shot learning not doable using just black box access to a distinguisher).
- Paper: `https://ia.cr/2019/037`
- Code: `https://www.github.com/agohr/deep_speck`

Thank You For Your Attention!

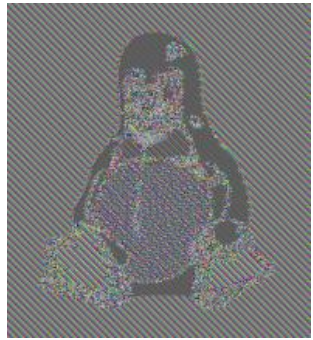


Figure: Breaking ECB mode encryption using *natural* neural networks. Image source: Wikipedia, *Block Cipher Mode of Operation*, accessed 2019-07-23. Original unencrypted image due to Larry Ewing, created using GIMP.

Contact

Bundesamt für Sicherheit in der Informationstechnik (BSI)
Dr. Aron Gohr
Godesberger Allee 185-187
53175 Bonn
Tel: +49 (0)228-9582-5969
Fax: +49(0)228-10-95825969
email: aron.gohr@bsi.bund.de