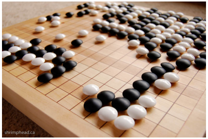
# Can AI Beat Cryptographers

David Gerault

University of Surrey

dagerault@gmail.com

# Introduction: Recent Feats of AI




Go: Alphago vs Lee Sedol, 2016

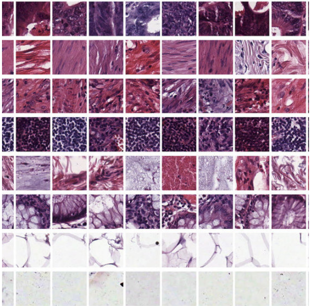
Poker: Libratus vs 4 pros, 2017

Chess: Deepblue vs Kasparov, 1997, AlphaZero (2017)

Starcraft 2: Alphastar grandmaster, 2019

# Introduction: More Feats

Lipnet: 93% vs 52%



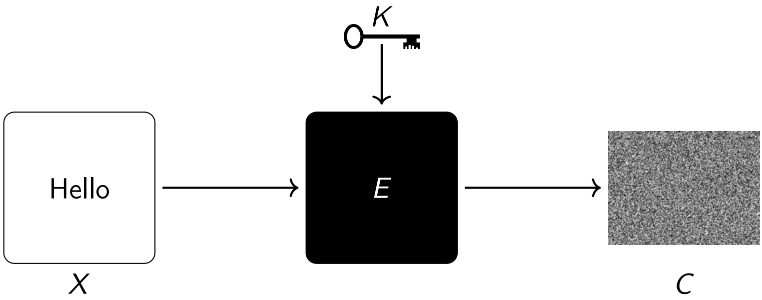
Cancer detection: Accuracy comparable to human specialists

# Can AI Beat Cryptographers?

- What does beating cryptographers even mean?
  - Creating ciphers?
  - Analysing ciphers?
- Assisting rather than beating?
  - Applying known attack strategies
  - Finding new attack strategies -> Interpretability

**But before... Some preliminaries!**

# What do I Mean by... Cryptography? (1)

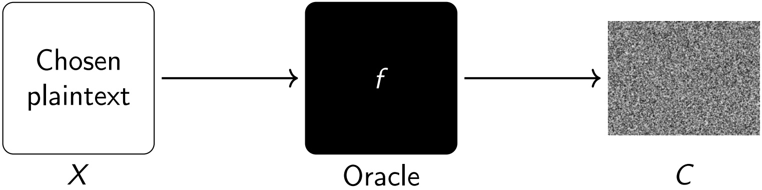



**Keyed permutation** $E\colon \{0,1\}^{\mathcal{K}} \times \{0,1\}^{\mathcal{P}} \rightarrow \{0,1\}^{\mathcal{P}}$**. Generally simple function iterated** $n$ **times.**

## Expected Property

Indistinguishable from a random permutation if $K$ is unknown

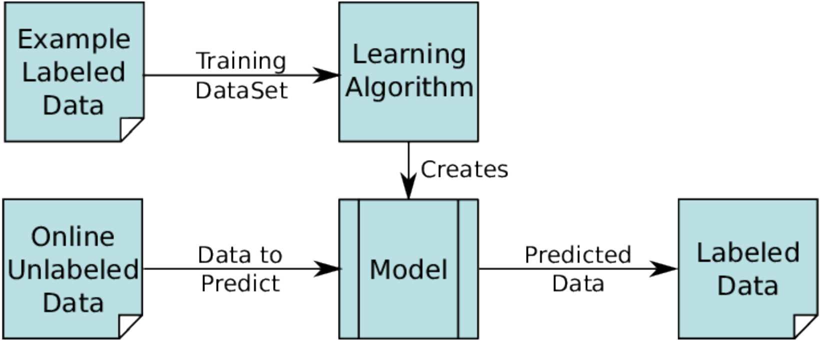
# What do I Mean by... Cryptography? (2)




$f \stackrel{?}{=} E_K$ **or random permutation** $\pi$**?**

**Distinguishing from** $\pi \equiv$ **recovering** $K$

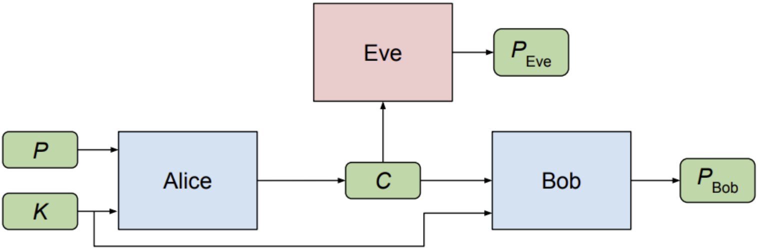
# What do I Mean by... AI



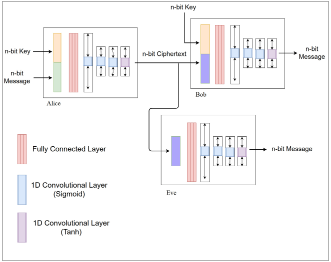
# Adversarial Neural Cryptography (ANC)

**Learning to Protect Communications with Adversarial Neural Cryptography**

M. Abadi, D. Andersen, 2016.



# ANC: Training Pipeline




- For $i \in [1, \text{nsteps}]$:
  - Alice and Bob train for $X$ iterations;
  - Eve trains for $2 \cdot X$ iterations;
- Sanity check: retrain Eve from scratch 5 times
- Success if decryption works, and Alice's advantage is no more than 2 bits.

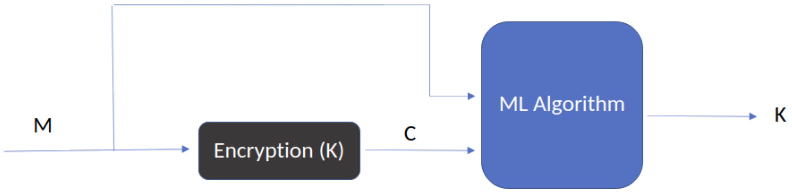
# Limitations of ANC

- Adversary [1]
- Infinite key material
- No simple expression
- Non-zero decryption error

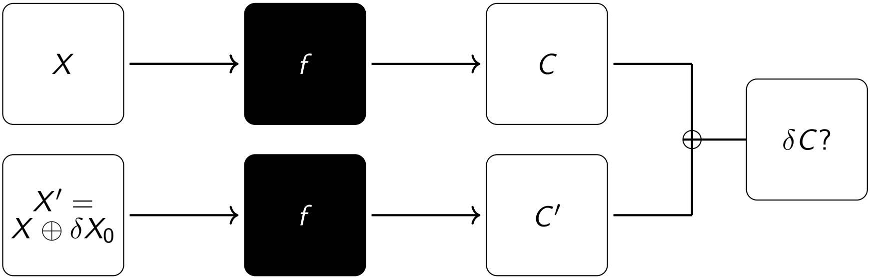
[1] **Learning Perfectly Secure Cryptography to Protect Communications with Adversarial Neural Cryptography**, M. Coutinho, R. Albuquerque, F. Borges, L. Villalba, T. Kim, SENSORS 2018

**Cryptographers beat AI (so far...)**

# AI Learning Cryptanalysis: How?



# A Word on Differential Cryptanalysis

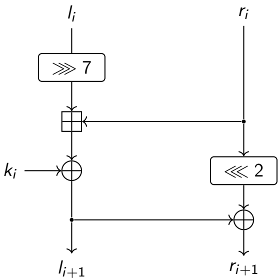



Distribution of $\delta C$ for a chosen $\delta X_0$...

If $f = \pi$ ? Uniform
If $f = E_K$ ? Not uniform!

# The SPECK Block Cipher

$$l_{i+1} = ((l_i \ggg 7) \boxplus r_i) \oplus k_i$$
$$r_{i+1} = (r_i \lll 2) \oplus l_{i+1}$$

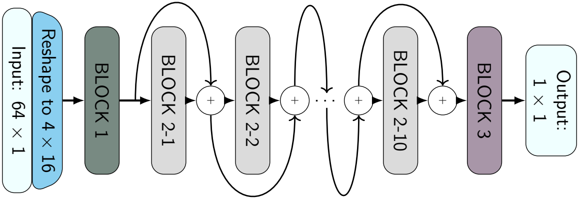



**In this talk...**

$$\delta_0 = (\delta L_0, \delta R_0) = (0x0040, 0x0000)$$
$$i.e., \delta L_0 >> 7 = 1000000000000000$$

# DeepSPECK

**Improving Attacks on Round-Reduced Speck32/64 using Deep Learning**, A. Gohr, CRYPTO 2019




| Input: $C0, C1$ | |
|---|---|
| Label 0 (Random) | Label 1 (SPECK) |
| $P0, P1$=Rand($2^{64}$)<br>$K$=Rand($2^{32}$)<br>$C0$=SPECK$_K(P0)$<br>$C1$=SPECK$_K(P1)$ | $P0$=Rand($2^{32}$), $P1 = P0 \oplus \delta$<br>$K$=Rand($2^{32}$)<br>$C0$=SPECK$_K(P0)$<br>$C1$=SPECK$_K(P1)$ |

# The Real Vs. Masked Experiment

| Input: $C0, C1$ | |
|---|---|
| Label 0 (SPECK Masked) | Label 1 (SPECK) |
| $P0$=Rand($2^{32}$), $P1 = P0 \oplus \delta$ | $P0$=Rand($2^{32}$), $P1 = P0 \oplus \delta$ |
| $K$=Rand($2^{32}$) | $K$=Rand($2^{32}$) |
| $M$=**Rand**($2^{32}$) | $C0$=SPECK$_K(P0)$ |
| $C0$=SPECK$_K(P0) \oplus M$ | $C1$=SPECK$_K(P1)$ |
| $C1$=SPECK$_K(P1) \oplus M$ | |

$$(C0 \oplus M \oplus C1 \oplus M = C0 \oplus C1)$$

**The NN learns something more than differences?**

# Gohr's Results

| | Normal Case | | | Random Vs Masked |
|---|---|---|---|---|
| Nr | Accuracy | TPR | TNR | Accuracy |
| 5 | 0.911 | 0.877 | 0.947 | 0.707 |
| 6 | 0.788 | 0.724 | 0.853 | 0.606 |
| 7 | 0.616 | 0.533 | 0.699 | 0.551 |
| 8 | 0.514 | 0.519 | 0.508 | 0.507 |

Remember that classification is performed with a single pair!

(Normal with only the differences as input: 0.9, 0.75, 0.58)

(+ improvement of the best 12 rounds key recovery on SPECK32, $2^{38}$ vs. $2^{46}$)

# Gohr's Key Recovery (Basic version)

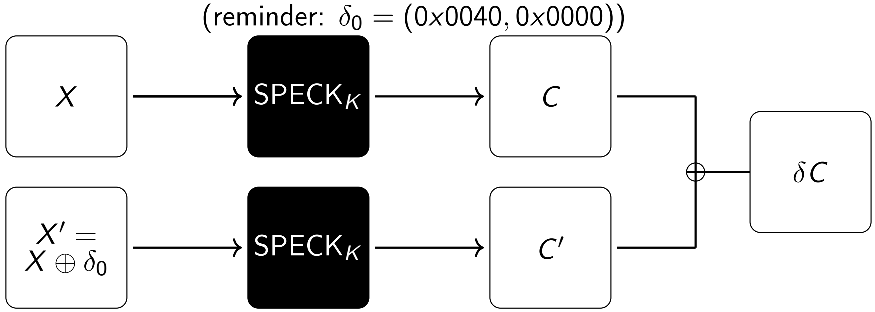
- For $k \in [0, 2^{16} - 1]$:
  - $X_0 = \text{decOneRound}(C_0,\ k)$
  - $X_1 = \text{decOneRound}(C_1,\ k)$
  - $\text{Scores}[k] = \text{N}(C_0, C_1)$
- Return indexOf(MAX(Scores))

# Our paper

**A Deeper Look at Machine Learning-Based Cryptanalysis**,
A. Benamira, D. Gerault, Q. Tan, T. Peyrin, Eurocrypt 2021

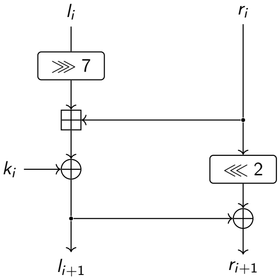
- Q1: Can we do better by hand?
- Q2: What does the NN learn? (Cryptanalysis aspect)
- Q3: What does the NN learn? (Interpretability)
- Extentions: SPN vs ARX
- Improving the accuracy

# Empirical Experiments




- No more than $n = 10^7$ pairs
- Prediction on a single pair
- $\text{DDT}[\delta C] = \frac{\#\{C \oplus C' = \delta C\}}{n}$
- **BAD!**: 0.73 accuracy for 5 rounds (Vs. 0.92)

# Refining the Experiments




- $\delta V_i = \delta L_i \oplus \delta R_i$: 0.85 for 5 rounds (Vs. 0.92)
- Individual difference bit biases: Still not 0.92
- Masking:
  - Let $M = M_L, M_R$ a fixed mask
  - aDDT$[\delta C \wedge M] = \frac{\#\{(C \oplus C') \wedge M = \delta C \wedge M\}}{n}$

# The Average Key Rank Distinguisher

- Compute aDDT
- For $k \in [0, 2^{16} - 1]$: //(Approximation)
  - $X_0 = \text{decOneRound}(C_0,\ k)$
  - $X_1 = \text{decOneRound}(C_1,\ k)$
  - Scores$[k]$ =aDDT$[(C \oplus C') \wedge M)]$
- Return Avg(Scores) $\geq 2^{-|M|}$

With M=$(0xff8f, 0xff8f)$...

| | **Gohr** | | | **aDDT** | | |
|---|---|---|---|---|---|---|
| Nr | Accuracy | TPR | TNR | Accuracy | TPR | TNR |
| 5 | 0.911 | 0.877 | 0.947 | **0.929** | 0.907 | 0.952 |
| 6 | **0.788** | 0.724 | 0.853 | **0.788** | 0.725 | 0.85 |
| 7 | **0.616** | 0.533 | 0.699 | 0.603 | 0.553 | 0.652 |
| 8 | **0.514** | 0.519 | 0.508 | N/A | N/A | N/A |

# Back to Gohr: Comparing Good and Bad Pairs

- Good pairs (G): NN score greater than 0.9
- Bad pairs (B): NN score lower than 0.1
- We are looking at round Nr−2 (3 or **4**, for Nr = 5, **6**)

| bit position | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| *G* | 0.476 | -0.454 | -0.355 | -0.135 | 0.045 | 0.084 | -0.009 | 0.487 | -0.473 | -0.426 | -0.300 | -0.050 | 0.006 | 0.019 | 0.500 | -0.500 |
| *B* | -0.002 | 0.018 | 0.008 | -0.011 | 0.044 | 0.002 | 0.023 | -0.022 | 0.010 | -0.002 | 0.013 | -0.004 | 0.006 | -0.005 | 0.103 | 0.072 |

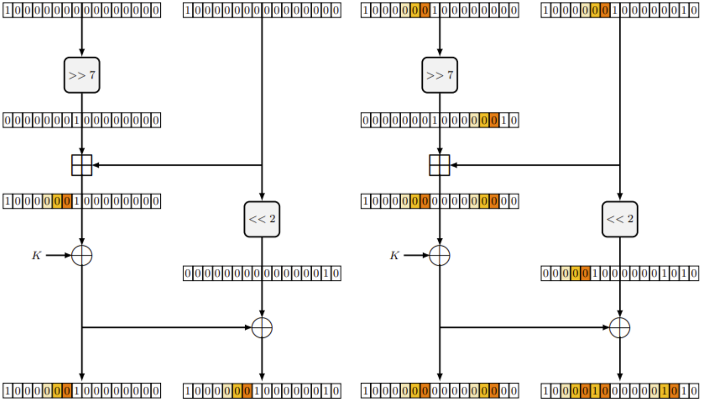
| bit position | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| *G* | 0.476 | -0.454 | -0.142 | -0.006 | 0.025 | 0.084 | -0.009 | 0.487 | -0.473 | -0.426 | 0.165 | 0.094 | -0.006 | 0.019 | -0.500 | -0.500 |
| *B* | 0.031 | -0.009 | -0.015 | -0.007 | -0.014 | -0.024 | 0.025 | 0.026 | 0.034 | -0.005 | -0.018 | -0.021 | 0.006 | 0.009 | 0.079 | -0.065 |

**Good pairs tend to follow this pattern!**

3 rounds: $10 \ast\ast\ast\ast\ast 00 \ast\ast\ast\ast\ast 00\ 10 \ast\ast\ast\ast\ast 00 \ast\ast\ast\ast\ast 10$

4 rounds: $10 \ast\ast\ast\ast\ast 10 \ast\ast\ast\ast\ast 10\ 10 \ast\ast\ast\ast\ast 10 \ast\ast\ast\ast\ast 00$

# Propagation of the Initial Difference




Round 1 -> 2 (left), 2 ->3 (right)

# Multiple Linear Approximations?

TD3: $10 * * * * * 00 * * * * * 00\ 10 * * * * * 00 * * * * * 10$

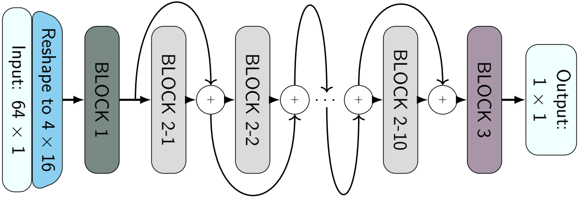
TD4: $10 * * * * * 10 * * * * * 10\ 10 * * * * * 10 * * * * * 00$

| Round | Trunc. Diff. | Dataset size | Acc. | Proport. |
|---|---|---|---|---|
| 3 | TD3 | 87741 | 0.992 | 87.11% |
| 4 | TD4 | 50063 | 0.999 | 50.06% |

**Differential-Linear Cryptanalysis with multiple linear approximations?**

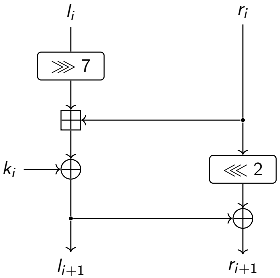
(But we still don't know what combinations of bits to look at)

# Dissecting the Neural Network



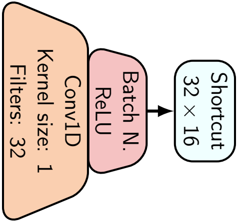

- Tweaking the inputs
- Deriving the features learnt

# Tweaking the Inputs




Hypothesis: $(\delta L, \delta R, V0, V1)$ works -> Confirmed!

# Interpreting the Outputs of Block 1




- Replace relu activation by heavyside to force binary output
- Train with inputs $\delta L, \delta R, V0, V1$
- Observe outputs of block 1
- $\delta L, \neg V0 \wedge V1, \neg\delta L, \neg V0 \wedge \neg V1, \delta L \wedge \delta V, \neg\delta L \wedge \neg\delta V$

# Extracting Relevant Masks Automatically

- Divide data into catgories ( G/B)
- Derive important bit for each categories (Captum)
- Combine these bits into masks on $X = (\delta L, \delta R, V0, V1)$
- Derive M-ODT(X, M) = $\Pr[X \wedge M | \text{SPECK}]$
- Replace the output of block 1 with M-ODT(X, $M_i$)

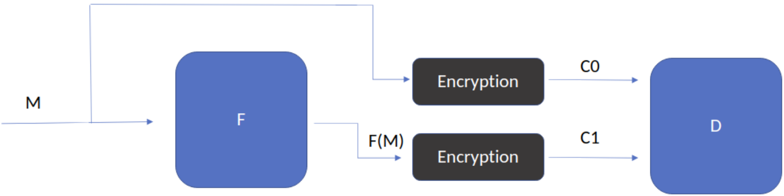
With 150 masks, and LGBM as a classifier, we almost reach Gohr's accuracy (-1%)

**Cryptographers + AI FTW!**

# Limits of this Approach

- Restricted to practical attacks
- Complexity analysis
- The NN is still guided
- (Maybe) not as efficient on SPN ciphers
- Still a lot to uncover!

# The Future of ML for Cryptanalysis?



# Conclusion: Can AI Beat Cryptographers?

- Cipher design: No
- Cipher analysis: Yes
- Through interpretability, AI may assist cryptographers

**Questions?**