University of BRISTOL

November 11, 2018

A Systematic Study of the Impact of Graphical Models on Inference-based Attacks on AES

Joey Green, Elisabeth Oswald, Arnab Roy

EPSRC Pioneering research and skills
erc European Research Council
National Cyber Security Centre a part of GCHQ

Outline of Talk

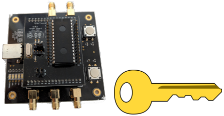
Threat Model

WE HAVE

- a copy of the real device
- source code of AES running on real device
- power traces from the real device
- plenty of compute power

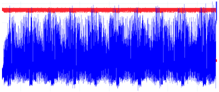
WE WANT

- the key used in the real device



WHILST MINIMISING

- number of power traces from real device



Same threat model as a Template Attack

What is Belief Propagation?

What is BP?

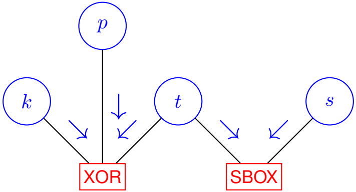


Figure: Variable Nodes send messages, edges updated

What is BP?

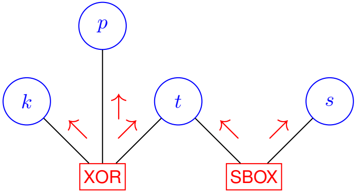


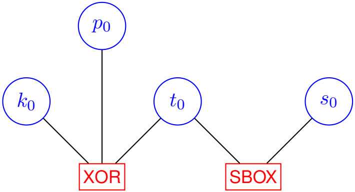
Figure: Factor Nodes send messages, edges updated

How we use BP

How we use BP I: Building the Factor Graph

```
ldrb r3, #key
ldrb r4, #plaintext
eors r3, r4
...
ldrb r4, [#sbox, r3]
```

$\Longrightarrow$



How we use BP II: Acquiring the ‘messages’

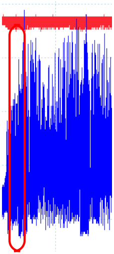


Figure: Timepoint where node x is computed

$\rightarrow$

256 TEMPLATES

(μ_0, σ_0)

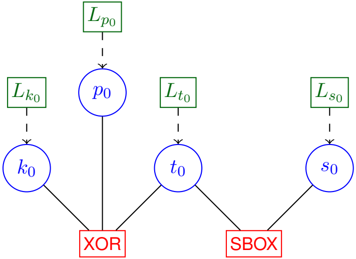
(μ_1, σ_1)

$\dots$

$(\mu_{255}, \sigma_{255})$

TEMPLATE MATCH $\Longrightarrow [0.01, 0.08, 0.02, \dots, 0.015]$ probability distribution size 256

How we use BP III: Performing the Attack



RUN
BP
$\Longrightarrow$

Compute Marginals of
all Key bytes
to get
Final Ranking of Key
between 1 and 2^{128}

- Marginal of $k_0 = L_{k_0} \cdot \text{message}_{\text{XOR} \rightarrow k_0}$

Previous Work

Previous Work

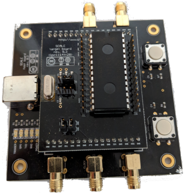
- Work in [Veyrat-Charvillon et al., 2014] demonstrated Belief Propagation Attack on AES
  - First step into practical attack, follow up work [Grosso and Standaert, 2015] showed improvement over DPA with enumeration
- We worked on this existing work because we noticed in some cases the attack can fail
  - Our data was very noisy
- We devised a number of improvements to get the attack to work on our data

Attack Setup

Attack Setup

TARGET DEVICE

- SCALE Host Board
- ARM Cortex m0
- 50MHz clock



OSCILLOSCOPE

- Picoscope 2000 Series
- 8 bit resolution at 500 MS/s

CODE TO RUN BPA

- Python 2.7
- cython, numpy, networkx
- On Github

Our improvements

Improvement I: Ending Early

- In literature, BP is run for a set number of iterations t_{max}
- In practice, message information may diffuse after a small number of iterations
- We propose an improvement that detects message diffusal, and terminates safely
- We call this Epsilon Exhaustion in the paper

Improvement II: Detecting Erroneous Traces

- In real scenarios, we may have a trace that provides erroneous information
- This might occur when there is a large influx of noise when acquiring the trace
- We propose an improvement to increase success rate by removing these
- Our improvement detects these by looking at the marginals of the plaintext bytes
- We call this the Ground Truth Check in the paper

Improvement III: Removing Nodes

- We propose an improvement to simplify the factor graph by omitting certain nodes, which simplifies complexity
- Our method selects nodes for omitting by calculating the ‘importance’ of each node
- Advantages:
  - If a leakage point is very noisy, removing this from the graph prevents noise propagation
  - Reduces the size of the graph, saving memory and compute time
- Disadvantages:
  - Reduces information provided to the BP algorithm
    - (but how much?)

Improvement III: Removing Nodes

- To calculate the ‘importance’ of node x:

Use standard leakage for node x and run BP, storing the final key distributions

$\Longleftrightarrow$ HELLINGER DISTANCE

Fix node x to have values 0 to 255; for each value, run BP, storing the final key distributions each time

- We calculated this distance for all non-key variable nodes in the Full AES Factor Graph

Improvement III: Removing Nodes

- Our observation: the more XOR nodes between a node and the key bytes, the less leakage information the node provides to the key bytes
- Example using the output of SubBytes:

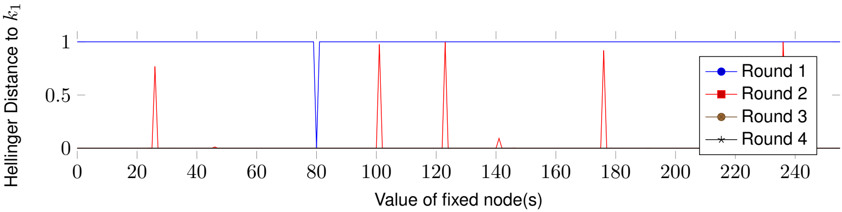


Figure: Hellinger Distance of k_1 to different fixed value s nodes (output of SubBytes) in AES rounds 1 to 4

Improvement III: Removing Nodes

- To get the 'best' possible factor graph for a given cryptographic algorithm, we want to remove nodes that are not important
- In our case: we only need to use the first two rounds of AES
  - This cuts our AES factor graph from 1212 variable nodes to 188
  - 84% size decrease with identical success rate

Improvement IV: Removing Cycles

- Belief Propagation works best when graph is acyclic
- But AES is a naturally cyclic algorithm
- When the graph has cycles, we call it Loopy BP
  - Loopy BP can exhibit peculiar behaviours, such as random oscillations of information
  - If Loopy BP fails to converge, the attack fails
- We propose an improvement by removing the cycles

Improvement IV: Removing Cycles

- Advantages:
  - Guaranteed convergence of BP
    - No more ‘unusual behaviour’
- Disadvantages:
  - Edges between nodes are severed, leading to loss of information
    - (but how much?)

Improvement IV: Removing Cycles

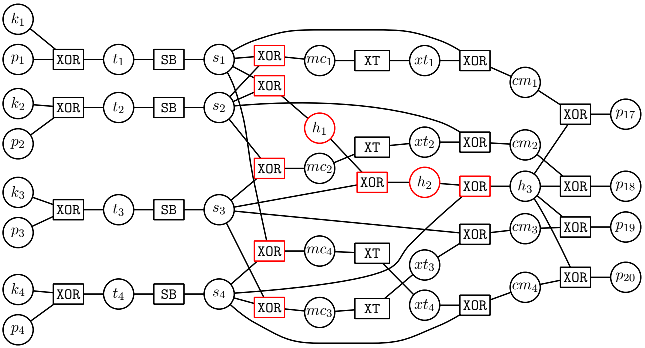


Figure: Cyclic Factor Graph G_1 representing the computation of a column in the first round of `AES FURIOUS`. The nodes in red can be removed to make the graph acyclic.

Improvement IV: Removing Cycles

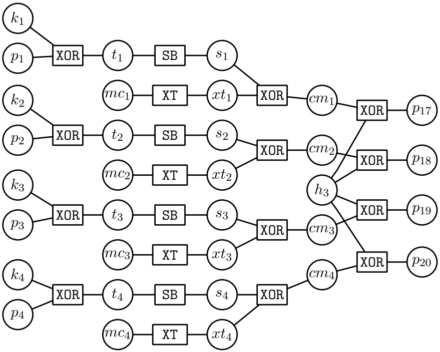


Figure: Acyclic Factor Graph G_1^A, computation of the first column in the first round of `AES FURIOUS`.

Improvement IV: Removing Cycles

- Full AES G, First Round Cyclic G_1, First Round Acyclic G_1^A
  - G_1 used for simplicity over G_2 as they perform similarly

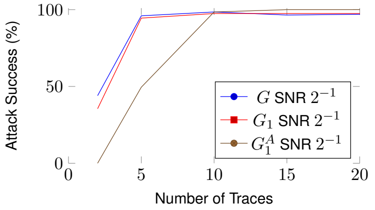


Figure: Graph Comparison, SNR = 2^{-1}

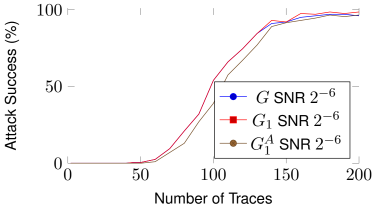


Figure: Graph Comparison, SNR = 2^{-6}

Improvement IV: Removing Cycles

Should you always remove cycles?

- In our case:
  - If the data is noisy, yes
  - We cut BP iterations t_{max} from 50 to 8
- In the general case:
  - Depends not only on SNR but on the structure of the graph
  - Definitely worth consideration

Improvement V: Connecting Multiple Graphs

- In cases of high noise (in most practical use cases), we require information from multiple traces to successfully recover the key
- We propose an improvement to combine information from multiple traces by performing BP on each trace independently

Improvement V: Connecting Multiple Graphs

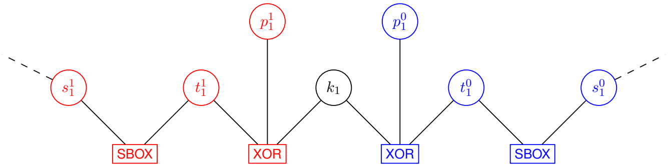


Figure: Connecting two (or more) traces to form a Large Factor Graph. The blue and red nodes correspond to two different factor graphs (traces) where the node k_1 is common to both of them

- Advantages: information from one trace can propagate into another
- Disadvantages: memory requirement scales with number of traces, cannot detect erroneous traces

Improvement V: Connecting Multiple Graphs

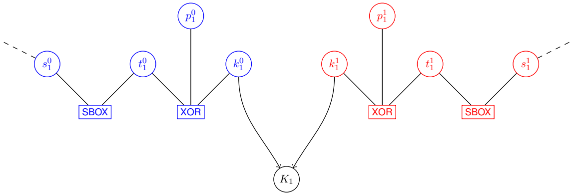


Figure: Two (or more) Independent Factor Graphs connected via a universal key node. The blue and red nodes correspond to two different factor graphs (traces) where the node k_1 in each trace connects to a universal key node K_1

- Advantages: memory requirement fixed size of one trace, parallelisable
- Disadvantages: information cannot propagate through other traces

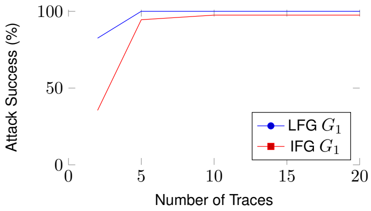


Figure: Cyclic Graph Connection Comparison, SNR = 2^{-1}

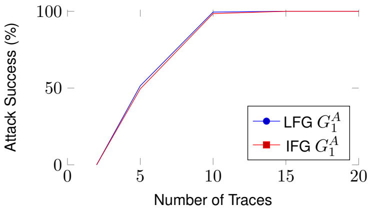


Figure: Acyclic Graph Connection Comparison, SNR = 2^{-1}

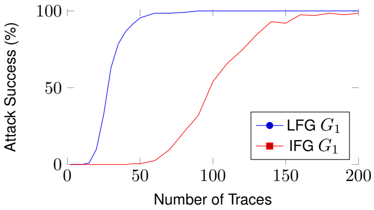


Figure: Cyclic Graph Connection Comparison, SNR = 2^{-6}

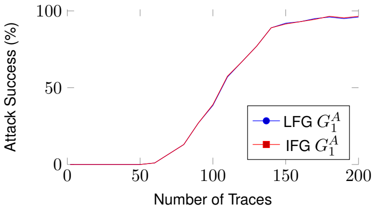


Figure: Acyclic Graph Connection Comparison, SNR = 2^{-6}

- Conclusion: depends on your attack setup, overall we suggest IFG

Combining Improvements

Combining Improvements

1. Removing nodes can be used alongside any improvement
2. Ending Early only works on cyclic graphs
3. Detecting Erroneous Traces only works on IFG

Conclusion

Conclusion

- Our proposal for practical use:
  - Always reduce to no more than 2 rounds of AES
  - If you have noisy traces or a more complex device, try removing bad traces using IFG method
  - Once removed, run attack with LFG and run attack with IFG
    - This is to find best configuration for your device
- Belief Propagation requires tuning more parameters than a standard template attack

Fin

`github.com/JustJoeyGreen/belief_propagation_attack`

Bibliography I

Grosso, V. and Standaert, F.-X. (2015).
ASCA, SASCA and DPA with enumeration: Which one beats the other and when?
pages 291–312.

Veyrat-Charvillon, N., Gérard, B., and Standaert, F.-X. (2014).
Soft analytical side-channel attacks.
In Sarkar, P. and Iwata, T., editors, *Advances in Cryptology – ASIACRYPT 2014*, pages 282–296, Berlin, Heidelberg. Springer Berlin Heidelberg.