МЕТОД АНАЛИЗА НЕ СИНХРОНИЗИРОВАННЫХ СИГНАЛОВ ПОБОЧНОГО КАНАЛА БЕЗ ПРЕДОБРАБОТКИ

Анисимов В.В.

Научный руководитель – зав.кафедрой САИТ, д-р техн. наук Латыпов Р.Х.

Казанский (Приволжский) федеральный университет, Институт вычислительной математики и информационных технологий.

Vlad9n116@gmail.com

Введение

В современном мире нас окружает множество систем, содержащих криптографический модуль. В задачу этого модуля входит обеспечение безопасности данных пользователя и его аутентификация. Безопасность данных пользователя часто строится на встроенном еще на этапе производства ключе шифрования внутри криптографического модуля, поэтому разработчикам таких систем необходимо предусмотреть всевозможные атаки на разрабатываемую систему и не дать злоумышленникам возможность получить доступ к секретному ключу. Одним из видов атак на криптографические системы являются атаки по сторонним (побочным) каналам. Идея данных атак строится на том, что криптографическая система для проведения операций шифрования и расшифровки манипулирует промежуточными значениями, которые связаны с секретным ключом тем или иным образом. В процессе работы система потребляет электроэнергию и производит электромагнитное излучение. Данные виды сигналов, производимые системой в результате работы зависят от данных, которыми манипулирует криптографическая система. Сигнал, полученный из системы во время ее работы называется сигналом побочного канала и является следствием физических процессов, происходящих внутри системы. Собирая и анализируя данные об побочных (сторонних) каналов злоумышленник может восстановить секретный ключ шифрования, в результате чего система будет скомпрометирована.

С развитием встраиваемых систем и методов защиты от утечек критически важных данных через побочный канал развивались и методы атак. Наиболее современные методы атак по сторонним каналам основаны на машинном обучении. Обычно такие атаки делят на два типа: «профильные» и «непрофильные». Для «профильных» атак характерно наличие дополнительного идентичного атакуемому устройства. Злоумышленник используя это устройство и контролируя ключ шифрования в нем может изучать и выявлять шаблоны в сигнале стороннего канала, а затем использовать их для извлечения секретного ключа из целевого устройства [1, 2]. К «непрофильным» атакам, соответственно, относят атаки в которых у злоумышленника есть лишь возможность собирать данные побочного канала с целевого устройства при разных входных данных [3].

В 2019 году B.Timon показал в возможность использования нейронных сетей для «непрофильных» атак [4]. Однако в этой работе не была показана возможность атаки на несинхронизированные данные для защищенной реализации алгоритма шифрования внутри встраиваемой системы. В данной работе будет показана архитектура нейронной сети, позволяющая производить атаку и на такие данные. Более того будет показана возможность локализации мест утечки информации из побочного канала с помощью анализа градиента.

Работы по данной теме

В своей работе B.Timon [4] разработал метод, который позволяет восстановить правильный ключ, обучая 256 нейронных сетей предсказывать промежуточные значения. Для

этого каждой нейронной сети соответствовал гипотетический фиксированный ключ, на основе которого для входного открытого текста вычислялось промежуточное значение, которым должна манипулировать система во время работы. В результате нейронная сеть, соответствующая правильному ключу предсказывала промежуточные значения с большей точностью чем все остальные. Однако для выполнения атаки необходима предобработка данных – необходимо вычислить среднее ожидаемое значение сигнала побочного канала в каждый момент времени и вычесть полученное значение из анализируемых сигналов. Данная операция выделяет отличия сигналов друг от друга, однако эффективно работает только в случае выровненных данных. В статье [4] была продемонстрирована успешная атака лишь на синхронизированные сигналы из набора данных ASCAD[5] с помощью многослойного персептрона (MLP), а атака на несинхронизированные данные с помощью сверточных сетей (CNN) осуществлялась только на незащищенную реализацию AES-128.

В 2022 году в работе [6] была представлена улучшенная модель для проведения атаки той же атаки что и в статье [4]. В новой архитектуре было предложено использовать для обучения лишь 1 модель с 256 выходами, соответствующие каждому гипотетическому ключу, что во много раз сокращает время для восстановления ключа. Однако в недостатках данной модели была указана невозможность восстановления мест утечки информации через побочный канал с помощью анализа градиента.

Разработанный метод

В новом разработанном методе предлагается предусмотреть в архитектуре нейронной сети слой синхронизации данных внутри пакета данных (batch). Используя слой MaxPooling после сверточного слоя возможно выравнивать извлеченные характеристики, как это показано на рисунке 1. За этим стоит следующая идея: сверточный слой, проходя своими фильтрами по сырым данным в пакете реагирует на выученные шаблоны и каждый выход является откликом на один и тот же фильтр. Используя слой MaxPooling возможно синхронизировать данные.

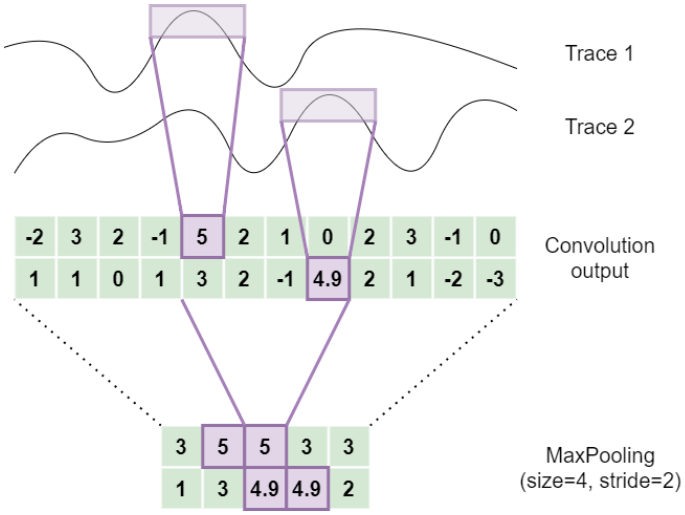

Рисунок 1 - Синхронизация данных с использованием MaxPooling

В результате, выходные значения операции MaxPooling будут некоторым образом синхронизированы. Далее необходимо «помочь» нейронной сети заметить небольшую разницу между данными внутри пакета. Для этого хорошим решением является использование операции BatchNormalization, представленной в 2015 году в статье [7]. Используя BatchNormalization нейронная сеть сама сможет вычислить статистику среднего внутри пакета

и использовать ее чтобы «увидеть» разницу между данными. Таким образом, с помощью данной последовательности слоев, возможно перенести предобработку данных в саму архитектуру нейронной сети, а также избавиться от необходимости синхронизации данных. Построенная модель одинаково хорошо работает как на выровненных данных, так и на сдвинутых относительно друг друга (несинхронизированных), однако необходимо чтобы окно MaxPooling было достаточно большим.

Для проведения атак в следующей главе были разработаны архитектуры CNN_{ASCAD} и CNN_{CHES}, которые можно найти в приложении.

Эксперименты

В данной главе будут показаны результаты атаки с помощью описанной выше архитектуры как на синхронизированные данные, так и на данные сдвинутые относительно друг друга. Все эксперименты проводились в Google Colab на процессоре Intel Xeon CPU 2.30GHz и видеокарте Tesla P100 16GB. Архитектура нейронной сети реализовывалась с помощью библиотеки PyTorch на языке Python3. В качестве целевых значений во всех экспериментах для обучения использовался младший бит промежуточного значения (lsb).

Первая атака проводилась на данные энергопотребления системы ChipWishper Lite, взятые с соревнований CHES 2016 CTF [8]. На вход нейронной сети CNN_{CHES} подавались первые 700 измерений со сдвигом (максимальный сдвиг 100). На рисунке 2 можно увидеть, как менялась точность предсказания целевого значения для каждого ключа с каждой эпохой обучения модели. Красная линия – точность предсказания значения для правильного ключа.

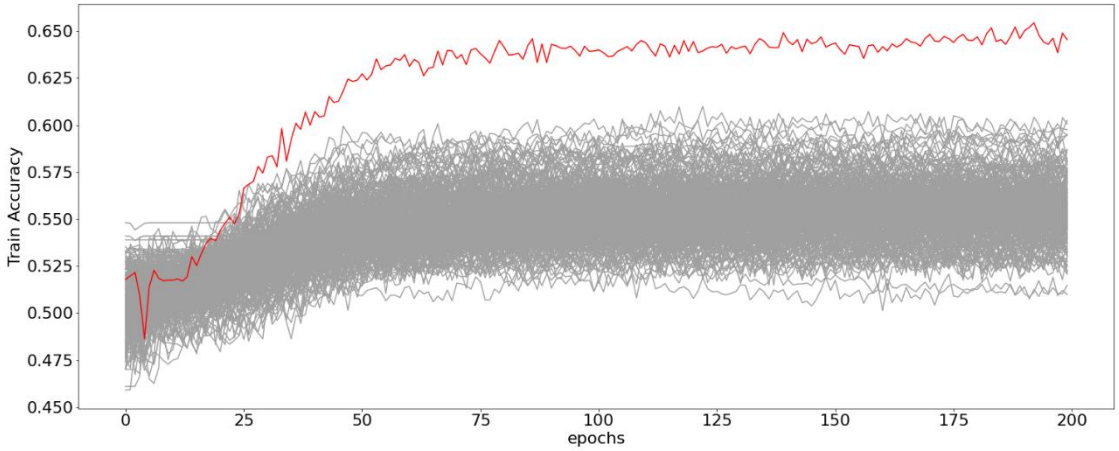

Рисунок 2 – Точность предсказания промежуточных значений для всех возможных ключей для набора данных CHES 2016

Более интересным примером является атака на открытый набор данных ASCAD [5]. Он был создан для сравнения моделей между собой и представляет из себя данные электромагнитного излучения собранные с ATMega8515, где в качестве алгоритма шифрования использовалась защищенная масками реализация алгоритма AES-128. Для демонстрации преимущества разработанной архитектуры CNN_{ASCAD} были проведены две атаки на набор данных ASCAD. На рисунке 3 представлена точность предсказывания промежуточных значения для несинхронизированных данных, а на рисунке 4 – для синхронизированных.

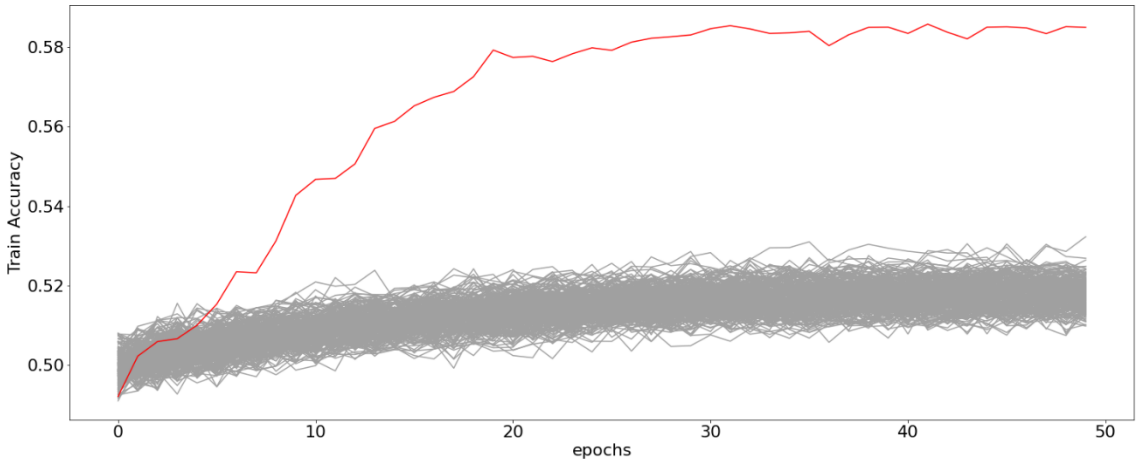

Рисунок 3 - Точность предсказания промежуточных значений для всех возможных ключей для набора несинхронизированных данных ASCAD

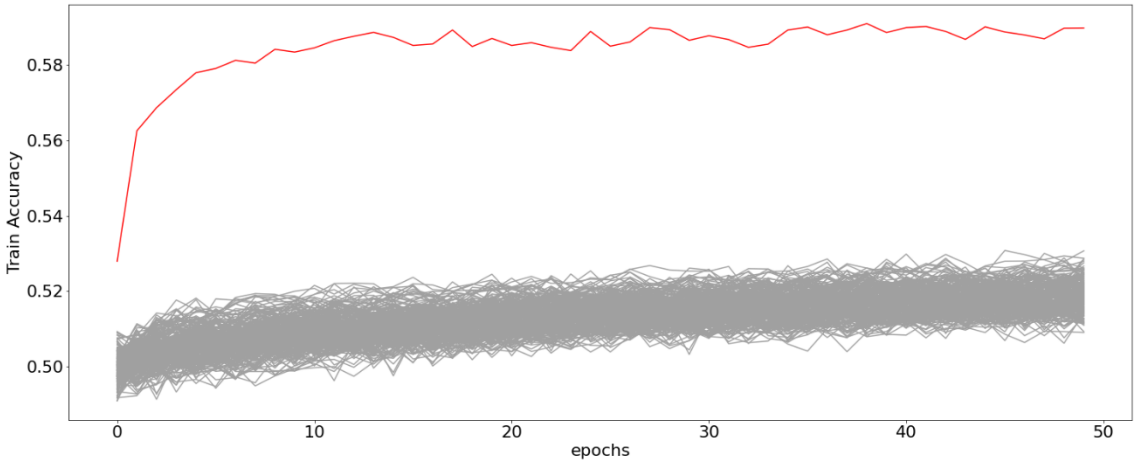

Рисунок 4 - Точность предсказания промежуточных значений для всех возможных ключей для набора синхронизированных данных ASCAD

Как видно из рисунков выше, модель после нескольких эпох обучения начинает предсказывать правильные значения для правильного ключа (красная линия) намного лучше, чем для неправильных.

Используя метод анализа чувствительности, описанный в статье [4], возможно провести анализ мест утечки информации. Однако из-за усовершенствованной архитектуры необходимо вычислять градиент только для выхода нейронной сети, который соответствует правильному предположению о ключе. Таким образом значение градиента на входе нейронной сети покажет на что именно реагировала нейронная сеть.

Для понимания того, на какие именно значения реагировала нейронная сеть необходимо вычислить коэффициенты корреляции Пирсона для различных промежуточных значений, используемых алгоритмом. Сопоставив полученные данные с значениями градиента возможно восстановить какие промежуточные значения важны нейронной сети для предсказания правильного ключа. На рисунке 5 можно увидеть сопоставленные значения градиента и коэффициентов корреляции для набора данных CHES 2016, а на рисунке 6 – для набора данных ASCAD[5].

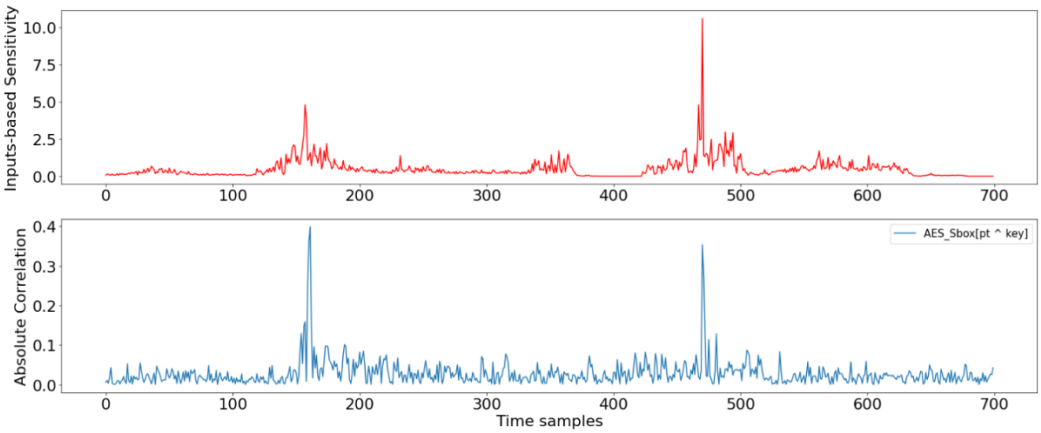

Рисунок 5 - Сопоставление коэффициентов корреляции Пирсона и градиента входов нейронной сети для набора данных CHES 2016

Из рисунка 5 можно сделать вывод, что в наборе данных CHES 2016 нейронная сеть реагировало на место утечки значения $Sbox(p_i \oplus k^*)$. Это объясняется тем, что в этом наборе данных используется классическая реализация AES-128 и, соответственно, промежуточное значение напрямую влияет на сигнал.

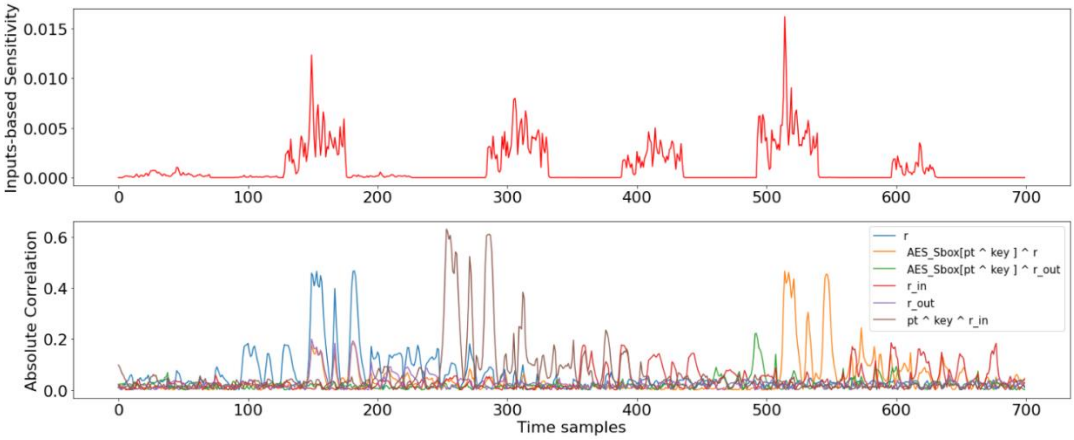

Рисунок 6 - Сопоставление коэффициентов корреляции Пирсона и градиента входов нейронной сети для набора данных ASCAD

Для набора данных ASCAD[5] используемый алгоритм AES-128 был модифицирован с целью защиты от атак по побочным каналам. Сделано это было с помощью использования масок во время всех операций, в результате чего нет прямой корреляции между промежуточным значением $Sbox(p_i \oplus k^*)$ и сигналами. Однако из рисунка 6 можно увидеть, что нейронная сеть реагировала на другие промежуточные значения, используемые модифицированным алгоритмом в процессе вычислений. В результате нейронная сеть смогла эффективно предсказывать значение $Sbox(p_i \oplus k^*)$, которое модифицированный алгоритм AES-128 не использует напрямую.

Заключение

В данной работе был предложен метод решения проблемы несинхронизированности данных в атаках по сторонним (побочным) каналам. Разработанный метод позволяет эффективно восстанавливать секретный ключ путем анализа сигналов побочного канала. Преимуществом данного метода является возможность работать с исходными данными без предварительной обработки. Также с помощью данного метода можно восстановить места утечки критической информации, которая может позволить злоумышленнику восстановить секретный

ключ. Локализация мест утечки и восстановление утекающих промежуточных значений даст возможность разработчикам криптографических систем улучшить систему и обезопасить конечного потребителя продукта.

ЛИТЕРАТУРА

1. S. Chari, J. R. Rao, P. Rohatgi Template attacks [Текст] / S. Chari, J. R. Rao, P. Rohatgi // Cryptograph. Hardw. Embedded Syst. – 2002. – С.13–28. –Режим доступа: https://doi.org/10.1007/3-540-36400-5_3 (дата обращения: 22.02.22).

2. W. Schindler, K. Lemke, C. Paar, A stochastic model for differential side channel cryptanalysis [Текст] / W. Schindler, K. Lemke, C. Paar // Cryptograph. Hardw. Embedded Syst. – 2005. – С.30–46. –Режим доступа: https://doi.org/10.1007/11545262_3 (дата обращения: 25.02.22).

3. E. Brier, C. Clavier, F. Olivier, Correlation power analysis with a leakage model [Текст] / E. Brier, C. Clavier, F. Olivier // Cryptograph. Hardw. Embedded Syst. – 2004. – С.16–29. –Режим доступа: https://doi.org/10.1007/978-3-540-28632-5_2 (дата обращения: 25.02.22).

4. B. Timon, Non-profiled deep learning-based side-channel attacks with sensitivity analysis [Текст] / B. Timon // Cryptograph. Hardw. Embedded Syst. – 2019. – С.107–131. –Режим доступа: https://doi.org/10.13154/tches.v2019.i2.107-131 (дата обращения: 07.03.22).

5. R. Benadjila, E. Prouff, R. Strullu, E. Cagli, C. Dumas, Deep learning for side-channel analysis and introduction to ASCAD database [Текст] / R. Benadjila, E. Prouff, R. Strullu, E. Cagli, C. Dumas // Journal of Cryptographic Engineering – 2020. – С.163–188. –Режим доступа: https://doi.org/10.1007/s13389-019-00220-8 (дата обращения: 07.03.22).

6. D. Kwon, S. Hong, H. Kim, Optimizing Implementations of Non-Profiled Deep Learning-Based Side-Channel Attacks [Текст] / D. Kwon, S. Hong, H. Kim // IEEE Access. – Т.10 – 2022. – С.5957–5967 – Режим доступа: https://doi.org/10.1109/ACCESS.2022.3140446 (дата обращения: 13.03.22).

7. Sergey Ioffe, Christian Szegedy, Batch normalization: Accelerating deep network training by reducing internal covariate shift [Текст] / Sergey Ioffe, Christian Szegedy // CoRR – T. abs/1502.03167 – 2015. – Режим доступа: https://doi.org/10.48550/arXiv.1502.03167 (дата обращения: 07.03.22).

8. CHES 2016 CTF dataset 57 [Электронный ресурс]. – 2016. – Режим доступа: https://ctf.newae.com/media/trace/traces57.7z (дата обращения: 07.03.22).

ПРИЛОЖЕНИЕ

CNN_{ASCAD}:

```
CNN(
  (net): Sequential(
    (0): Sequential(
      (0): Conv1d(1, 4, kernel_size=(33,), stride=(1,), bias=False)
    )
    (1): Sequential(
      (0): MaxPool1d(kernel_size=100, stride=100, padding=0, dila-
tion=1, ceil_mode=False)
      (1): BatchNorm1d(4, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (2): LeakyReLU(negative_slope=0.01)
    )
    (2): Sequential(
      (0): Conv1d(4, 32, kernel_size=(6,), stride=(1,), bias=False)
      (1): BatchNorm1d(32, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (2): LeakyReLU(negative_slope=0.01)
    )
    (3): Sequential(
      (0): Conv1d(32, 256, kernel_size=(1,), stride=(1,))
      (1): Sigmoid()
    )
  )
  (loss_fn): MSELoss()
)
```

CNN_{CHES}:

```
CNN(
  (net): Sequential(
    (0): Sequential(
      (0): Conv1d(1, 8, kernel_size=(32,), stride=(1,), bias=False)
    )
    (1): Sequential(
      (0): MaxPool1d(kernel_size=100, stride=50, padding=0, dilation=1,
ceil_mode=False)
      (1): BatchNorm1d(8, eps=1e-05, momentum=0.1, affine=True, track_run-
ning_stats=True)
      (2): LeakyReLU(negative_slope=0.01)
    )
    (2): Sequential(
      (0): AvgPool1d(kernel_size=(12,), stride=(12,), padding=(0,))
    )
    (3): Sequential(
      (0): Conv1d(8, 256, kernel_size=(1,), stride=(1,))
      (1): Sigmoid()
    )
  )
  (loss_fn): MSELoss()
)
```