УДК 519.876.5

Г. А. Мартвель, Ф. М. Чупраков, К. А. Недостоев, Н. С. Барыбин

Московский физико-технический институт (национальный исследовательский университет)

Программная реализация физически неклонируемых функций

Рассматривается задача моделирования физически неклонируемых функций на основе схемы на полевых транзисторах. Предлагается метод программной интерпретации задержек прохождения сигнала через физическую систему и делается оценка стабильности и устойчивости полученных моделей.

Ключевые слова: физически неклонируемые функции, защита информации, машинное обучение, арбитр, логистическая регрессия, аналоговая электроника.

G. A. Martvel, F. M. Chuprakov, K. A. Nedostoev, N. S. Barybin

Moscow Institute of Physics and Technology

Software implementation of physically unclonable functions

The problem of physically unclonable functions based on field effect transistors modeling is investigated in this research. The method of programming the interpretation of a delay connected with a signal passing through the physical system is suggested. The stability and sustainability estimation of an obtained model is provided.

Key words: physically unclonable functions, information security, machine learning, arbiter, logistic regression, analogue electronics.

1. Основные сведения о физически неклонируемых функциях

Определение 1. *Физически неклонируемая функция* (*ФНФ*) — некоторая функция, применяемая с помощью физической системы или ее модели, обладающая следующими свойствами:

- однородность выходных данных при одинаковых входных данных для одной реализации ФНФ;
- уникальность выходных данных для разных реализаций ФНФ;
- неклонируемость;
- однонаправленность (невозможность по выходным данным восстановить входные).

Называемая функцией, ФНФ не является таковой в строгом смысле. Поданный на вход стимул может быть поставлен в соответствие нескольким различным откликам, полученным на выходе. Это свойство объясняется неконтролируемыми погрешностями при производстве и различными видами шумов при работе аналогового устройства — реализации ФНФ. В качестве простейшей аналогии можно привести резисторы, для которых реальное сопротивление всегда отличается от заявленного производителем на какую-то случайную

величину. Именно этим обусловлена неклонируемость ФНФ: для двух реализаций с номинально одинаковой конструкцией реакция на одинаковые входные данные может значительно различаться. Для различных конфигураций ФНФ пропадает корреляция откликов и каждая из них обеспечивает уникальность выходных данных.

В рамках одной функции на различие выходных значений влияют только шумы. При реализации физической модели необходимо обеспечить условия, при которых разброс откликов не был бы слишком большим, т. е. их однородность. Это достигается подбором характеристик составляющих ФНФ элементов или регулированием внешних параметров, например, температуры среды.

Однонаправленность работы ФНФ реализуется непосредственно с помощью конструкции самой модели, поэтому будет рассмотрена на конкретном примере позднее.

Существует большое количество реализаций физически неклонируемых функций, различающихся используемыми материалами и технологиями производства. Так, выделяют ФНФ на полевых транзисторах, интегральных микросхемах, на оптических, магнитных или кремниевых элементах [1]. Во всех перечисленных типах ФНФ могут быть использованы различные способы генерации отклика на основе входного сигнала. Мы остановимся на схеме, работающей по принципу арбитра, реализованной с помощью полевых транзисторов.

2. Описание модели

Принцип действия физически неклонируемой функции, работающей по принципу арбитра (АФНФ), заключается в оценке задержки распространения сигнала, проходящего через элементы цепи, изображенной на рис. 1.

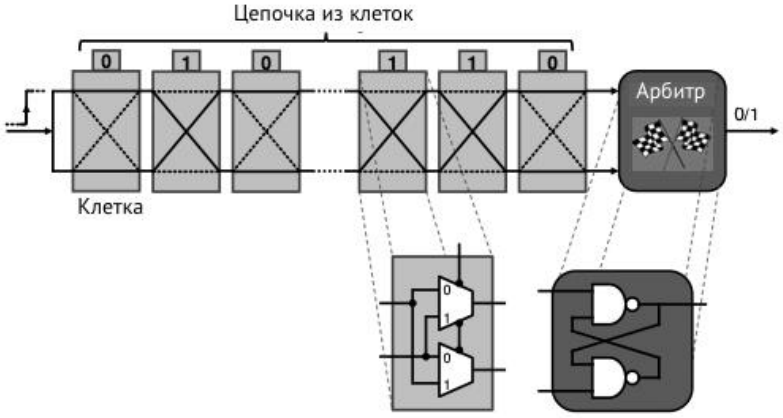

Рис. 1. Устройство АФНФ

Цепочка представляет собой последовательное соединение клеток, через которые по двум путям проходит сигнал. Клетка в свою очередь состоит из двух мультиплексоров, реагирующих на значение поданного бита. Например, если бит равен единице, сигналы проходят по клетке по прямому пути, если нулю, то сигналы меняются местами. Набор битов, по длине равный размеру цепи, является для АФНФ входным сигналом.

Мультиплексоры выполнены на полевых транзисторах, которые обеспечивают задержку сигнала. Именно ее флуктуации являются свойством, определяющим уникальность АФНФ. Каждый отрезок пути, по которому проходит сигнал, обладает своим собственным фиксированным временем задержки. В среднем, конечно, характеристики транзисторов соответствуют параметрам, заявленным производителем, поэтому, в силу влияния большого количества различных факторов, распределение случайной величины задержки можно принять за нормальное [2].

Таким образом, суммарное время прохождения на верхнем и нижнем путях будет отличаться на величину, определяемую последовательностью входных битов и шума, вызванного внешними факторами.

С учетом малости шумов, существует 2^n различных конфигураций суммарной задержки, где n — количество клеток. Это дает возможность атаковать систему методами машинного обучения [3], что будет выполнено позднее.

Выходным сигналом для АФНФ является бит, генерируемый арбитром — RS защелкой. В зависимости от того, по какому пути пришел первый сигнал, на выход подается ноль или единица. Тем самым реализуется однонаправленность реальной АФНФ, т. е. невозможность восстановления входной последовательности битов по одному выходному. При накоплении большого количества пар запрос-ответ такая возможность появляется.

3. Программная реализация модели

Определение 2. *Класс ФНФ* — полное описание конкретной реализации ФНФ.

Определение 3. *Объект класса ФНФ* — отдельно взятая ФНФ, сконструированная в соответствии с описанием класса ФНФ.

Понятие класса ФНФ можно легко представить в терминах объектно-ориентированного программирования. Для конкретного класса ФНФ (в нашем случае АФНФ) требуется реализовать методы *«создать»* и *«измерить»*. Метод «создать» будет конструировать объект заданного класса, а метод «измерить» будет возвращать отклик ФНФ на конкретный стимул.

Ниже приведена программная реализация АФНФ на языке программирования Python (рис. 2).

```python
import random

class Cell:
    def __init__(self):
        self.Ra = random.gauss(1,0.01)
        self.Rb = random.gauss(1,0.01)
        self.Rc = random.gauss(1,0.01)
        self.Rd = random.gauss(1,0.01)
    def cellTimeCounter(self, byte):
        if byte == 1:
            return self.Ra, self.Rd
        if byte == 0:
            return self.Rb, self.Rc

cells_num = 64

class APUF:
    def __init__(self):
        self.cells = [Cell() for i in range(cells_num)]
        self.cells_num = cells_num
    def evaluate(self, byte_arr):
        time_top = 0
        time_bottom = 0
        for i in range(self.cells_num):
            time_top += self.cells[i].cellTimeCounter(byte_arr[i])[0]
            time_bottom += self.cells[i].cellTimeCounter(byte_arr[i])[1]
        delta = time_top - time_bottom
        if abs(delta) < 0.01:
            delta += random.gauss(0,0.02)
        if delta > 0:
            return 1
        else:
            return 0
```

Рис. 2. Реализация класса АФНФ

Класс «Cell» («клетка») выступает в роли базового класса для построения класса АФНФ. В клетке, которая представляет собой абстракцию мультиплексора, реализованы 4 возможных пути распространения сигнала, для которых значения задержки распределены по Гауссу. АФНФ получается путем объединения 64 клеток. Метод «evaluate» («измерить») вычисляет разницу по времени в распространении двух сигналов и, в зависимости от этой разницы возвращает 0 или 1. Также в этом методе смоделирована ситуация, когда задержка между сигналами слишком мала: в этом случае компаратор выдает рандомизированное значение.

Определение 4. *ФНФ-эксперимент* — массив, состоящий из (N_{puf}, N_{chal}, N_{meas}) элементов, где N_{puf} — количество объектов ФНФ, N_{chal} — количество различных входных 64-битных последовательностей, N_{meas} — количество измерений для каждого входного значения, и полученный в результате измерения откликов N_{puf} объектов ФНФ на N_{chal} случайных стимулов, повторенных N_{meas} раз для каждого объекта ФНФ.

Чтобы прояснить написанное выше, приведем схему эксперимента для программной реализации АФНФ (рис. 3).

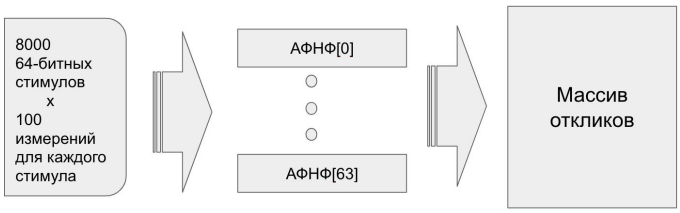

Рис. 3. Схема эксперимента

Как только массив данных собран, на его основе можно вычислить две важнейшие метрики, которые определяют, обладает ли данная программная реализация свойствами ФНФ.

Определение 5. *Интра-дистанция ФНФ* — случайная величина, описывающая дистанцию между двумя откликами одного и того же объекта ФНФ, полученных в результате действия одного и того же стимула,

$$D_{intra} \equiv \mathrm{dist}\left(Y(x), Y'(x)\right),$$

где x — стимул, $Y(x), Y'(x)$ — пара случайных выбранных откликов, полученных в результате двух различных измерений, $\mathrm{dist}(\cdot)$ — Хэммингово расстояние (в данном случае 0 или 1, т. к. каждая ФНФ выдает отклик длиной 1 бит).

Для всех возможных пар откликов строится массив интра-дистанций:

$$D_{intra}Arr = \left[\mathrm{dist}\left(Y_i^{(j_1)}(x_k), Y_i^{(j_2)}(x_k)\right)\right]_{1\le i\le N_{puf}, 1\le k\le N_{chal}, 1\le j_1\ne j_2\le N_{meas}}.$$

Затем для такого массива вычисляется математическое ожидание

$$\mu_{intra} = \frac{2}{N_{puf}\cdot N_{chal}\cdot N_{meas}\cdot (N_{meas}-1)}\cdot \sum D_{intra}Arr.$$

На рис. 4 представлена гистограмма распределения значений μ_{intra} для 1000 случайных стимулов. Эта метрика характеризует долю ошибочных откликов для конкретного стимула, в идеале это значение должно равняться нулю. В нашем эксперименте $\mu_{intra} = 0{,}03$, что в точности соответствует аналогичной метрике, измеренной в реальном эксперименте с физической реализацией АФНФ [4].

Теперь рассмотрим вторую ключевую метрику для анализа ФНФ.

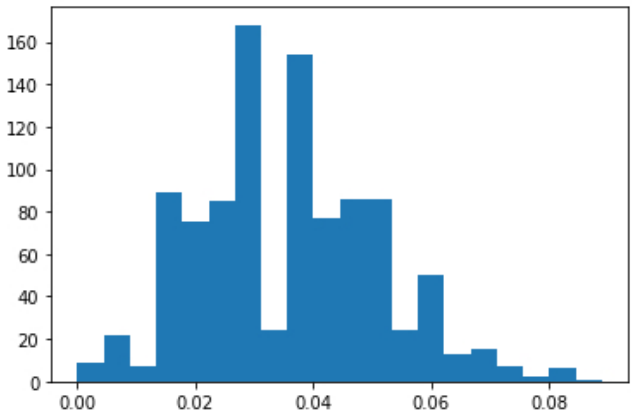

Рис. 4. Гистограмма μ_{intra}. По горизонтали — значения μ_{intra}, по вертикали — частота появления конкретного значения

Определение 6. *Интер-дистанция ФНФ* — случайная величина, описывающая дистанцию между двумя откликами ФНФ двух разных объектов ФНФ, полученных как результат одного и того же стимула,

$$D_{inter} \equiv \text{dist}\left(Y(x), Y'(x)\right),$$

где x — стимул, $Y(x), Y'(x)$ — пара случайных выбранных откликов, полученных в результате двух различных измерений на двух различных объектах ФНФ, $\text{dist}(\cdot)$ — Хэммингова дистанция.

Точно так же, как и в случае интра-дистанции, строится массив:

$$D_{inter}Arr = \left[\text{dist}\left(Y_{i_1}^{(j)}(x_k), Y_{i_2}^{(j)}(x_k)\right)\right]_{1\le i_1\ne i_2\le N_{puf}, 1\le k\le N_{chal}, 1\le j\le N_{meas}}.$$

Аналогично вычисляется математическое ожидание:

$$\mu_{inter} = \frac{2}{N_{puf}\cdot(N_{puf}-1)\cdot N_{chal}\cdot N_{meas}}\cdot\sum D_{inter}Arr.$$

На рис. 5 представлена гистограмма μ_{inter} для 1000 случайных импульсов. Эта метрика характеризует уникальность откликов различных объектов ФНФ на одинаковых стимулах. В идеале μ_{inter} должно быть равно 0,5: это бы означало, что два случайно выбранных отклика от разных ФНФ равновероятно либо отличаются, либо одинаковы. Для нашей модели $\mu_{inter} = 0,49$, что является очень хорошим результатом.

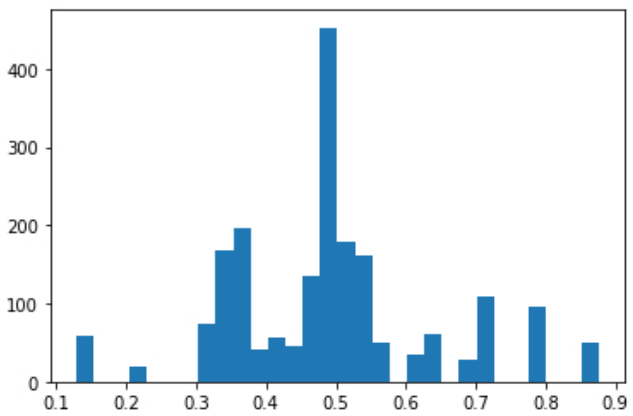

Рис. 5. Гистограмма μ_{inter}. По горизонтали — значения μ_{inter}, по вертикали — частота появления конкретного значения

Теперь пришло время дать пояснения на предмет того, каким образом данные метрики характеризуют ФНФ. Любая ФНФ должна обладать как минимум следующими шестью свойствами [4].

Определение 7. Говорят, что класс ФНФ обладает *конструктируемостью*, если существует метод, позволяющий относительно легко создать новый случайный объект класса ФНФ.

В реальности это определение означает, что существует технологический процесс, который позволяет реализовать объект класса ФНФ. В случае нашей программной реализации это свойство гарантируется наличием метода «init» («инициализировать») класса «APUF» («АФНФ»).

Определение 8. Говорят, что класс ФНФ обладает *измеримостью*, если для каждого объекта этого класса можно однозначно измерить отклик.

Разумеется, нет никакого смысла проектировать ФНФ, отклик которых нельзя измерить или однозначно интерпретировать. В нашем случае это свойство гарантируется наличием метода «evaluate» («измерить»).

Определение 9. Говорят, что класс ФНФ обладает *воспроизводимостью*, если он обладает измеримостью и относительно высока вероятность того, что интрадистанция мала.

Определение 10. Говорят, что класс ФНФ обладает уникальностью, если он обладает измеримостью и относительно высока вероятность того, что интердистанция большая.

Как видно, эти определения не являются в полной мере математически строгими, но зачастую на практике эти свойства гарантируются тем, что $\mu_{intra} \ll \mu_{inter}$, что в полной мере выполняется для нашей программной реализации.

Определение 11. Говорят, что класс ФНФ обладает *идентифицируемостью*, если он обладает уникальностью и воспроизводимостью.

Свойство идентифицируемости автоматически следует из свойств воспроизводимости и уникальности.

Определение 12. Говорят, что класс ФНФ обладает *физической неклонируемостью*, если он обладает измеримостью и сложно каким-либо образом повлиять на процедуру создания объекта класса таким образом, чтобы с высокой вероятностью получить два объекта, для которых значения интрадистанции и интердистанции существенно отклоняются от нормы, т. е. $\mu'_{inter} \ll \mu_{inter}$ и $\mu'_{intra} \gg \mu_{intra}$.

С практической точки зрения это означает, что крайне сложно повлиять на процесс создания объекта класса таким образом, чтобы получить два объекта с наперед заданными свойствами. В случае нашей программной реализации физическая неклонируемость гарантируется приватностью метода «init» («инициализировать»). Другими словами, мы никак не можем извне повлиять на работу этого метода.

Таким образом, мы показали, что программная реализация АФНФ обладает всеми необходимыми требованиями, которые предъявляются к ФНФ. Стоит также отметить, что преимуществами этой модели являются простота и интуитивность реализации: при моделировании использовались логичные и понятные приближения реальных физических процессов, а вычисленные метрики совпадают с реальными физическими аналогами АФНФ, описанными в [4].

4. Реализация атак

Стоит заметить, что предложенная реализация является линейной по последовательности входных битов. Действительно, каждому биту будет соответствовать определенный вес — значение задержки, а значение бита на выходе будет определяться суммой задержек на всех клетках.

Как было замечено ранее, линейные системы хорошо аппроксимируются линейными методами машинного обучения (мы будем использовать логистическую регрессию). Непредсказуемость ответов возникает только из-за неизвестных параметров системы, но, после обучения модели, каждый бит ответа легко прогнозируется с высокой точностью.

В качестве данных для обучения будем предоставлять моделям пары запрос-ответ — «случайная входная битовая последовательность-выходной бит». В качестве метрики точности выберем точность предсказания моделями задержек-весов для цепи, входящей в АФНФ.

Вследствие малой устойчивости реализации АФНФ к методам МО была предложена модификация, создающая нелинейность на выходе системы и повышающая ее надежность.

Поставим XOR (исключающее *«или»*) на выходе двух экземпляров АФНФ. Тогда значение бита на выходе не будет определяться линейной комбинацией задержек внутри одной АФНФ, а поэтому линейная модель машинного обучения будет менее точно предсказывать поведение такой системы. Таким образом можно объединять и более двух АФНФ, ставя на выходы полученных ансамблей новые XOR. Методы машинного обучения в этом случае будут воспринимать такой ансамбль, как эквивалентную линейную АФНФ.

В качестве другой модели МО будем использовать искусственную нейронную сеть (ИНС).

Проведем тестирование систем с разным количеством XOR. На рис. 6 представлен график зависимости точности предсказания задержек-весов АФНФ логистической регрессией от количества ансамблей.

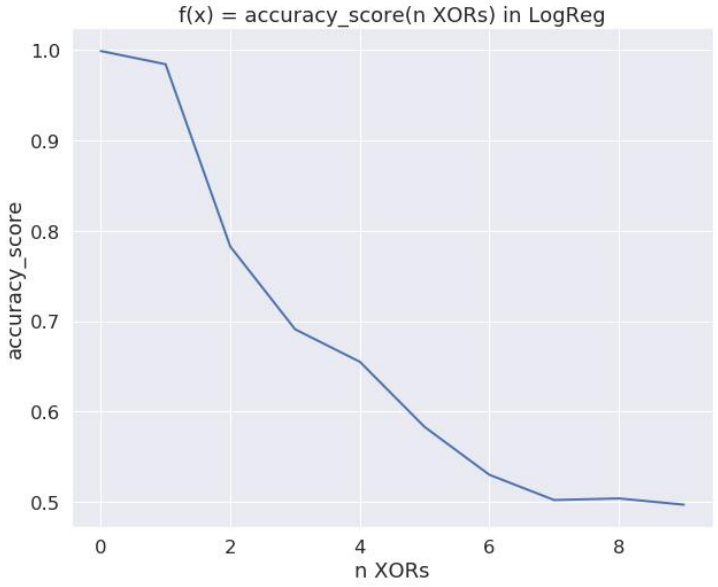

Рис. 6. Зависимость точности предсказания логистической регрессией от количества XOR

Из графика видно, что с ростом сложности модели точность предсказания падает и при семи и более XOR логистическая регрессия просто угадывает веса модели.

Для ИНС подобная зависимость представлена на рис. 7.

ИНС показывает схожий результат на том же наборе входных данных, переставая точно предсказывать задержки модели уже при пяти XOR.

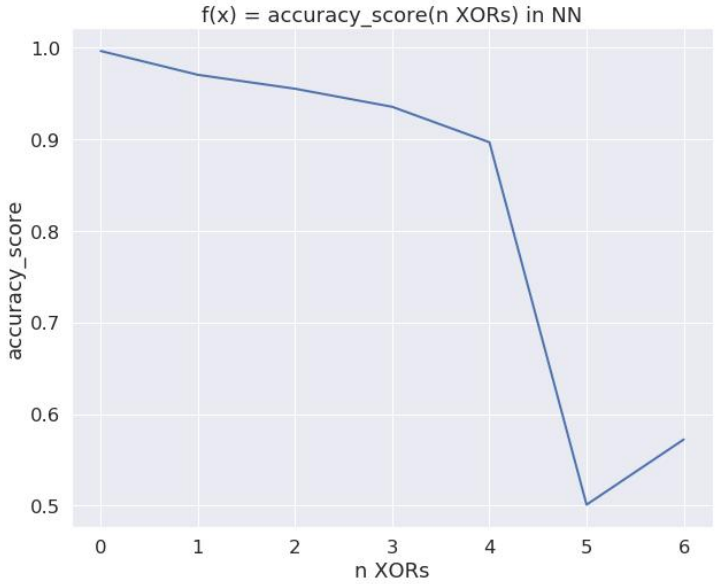

Рис. 7. Зависимость точности предсказания ИНС от количества XOR

Тем не менее на меньшем количестве нелинейных элементов ИНС сохраняет высокую точность, в отличие от логистической регрессии.

Очевидно, что чем больше обучающих примеров разрешено использовать алгоритму машинного обучения, тем выше становится точность его работы. Теперь попробуем увеличивать размер входного массива данных с целью повысить точность работы наших моделей.

Для логистической регрессии размер обучающей выборки не влияет на точность предсказания (с увеличением количества примеров точность для сложных систем все такая же низкая), а вот для ИНС влияет значительно, что отображено на рис. 8.

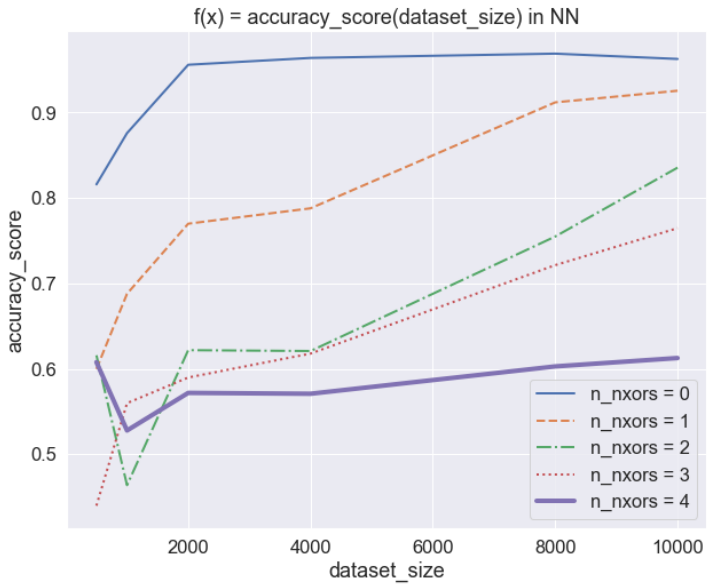

Рис. 8. Зависимость точности предсказания ИНС от размера обучающей выборки для разного количества ансамблей

Рост точности, вызванный увеличением количества входных данных, хорошо заметен на первых трех нелинейных реализациях АФНФ (с количеством XOR 1, 2 и 3 соответственно), а для более сложных моделей несуществен.

На практике получение большой обучающей выборки пар запрос-ответ предстает довольно трудоемкой задачей, поэтому хорошая (более 80%) точность предсказания для ре-

альной модели с несколькими нелинейными элементами не гарантирована. При этом стоит заметить, что подобное увеличение выборки требует больших вычислительных ресурсов и влечет за собой значительное увеличение времени обучения. С увеличением величины шума точность падает еще сильнее, что приведено в подобных исследованиях [4].

Таким образом, реализованная АФНФ с предложенными модификациями показала хорошую устойчивость к методам машинного обучения даже на относительно большом объеме данных. Модели, состоящие из пяти и более ансамблей параллельных АФНФ, объединенных элементами XOR, не подвержены аппроксимациям машинным обучением, при этом сохраняя все свойства ФНФ.

5. Выводы и дальнейшая работа

В ходе работы была предложена модель физически неклонируемой функции, являющаяся реализацией реальной аналоговой модели. Были исследованы свойства физически неклонируемых функций и осуществлена проверка соответствия полученной модели этим свойствам. Также с помощью методов машинного обучения были реализованы атаки на модель и была предложена конфигурация, устойчивая к различным типам моделирования.

Важной задачей представляется создание и усложнение модификации модели, позволяющей уменьшить затраты ресурсов на ее реализацию и увеличить устойчивость к атакам. Другой задачей является обеспечение защищенности программной реализации и повышение ее функционала с целью дальнейшего использования в различных проектах.

Литература

1. https://staff.aist.go.jp/hori.y/en/puf/index.html. Physically Unclonable Function. Tokio (Japan): National Institute of Advanced Industrial Science and Technology, 2011.
2. https://helpiks.org/9-21094.html. Теоретические законы распределений погрешностей параметров качества в производстве электронных средств, 2014.
3. *Ahmed Ferozpuri.* Modeling delay-based PUFs with Machine Learning. Фэрфакс (США): GMU Electrical and Computer Engineering, 2015.
4. *Roel Maes.* Physically Unclonable Functions: Constructions, Properties and Applications. Leuven (Belgium): Catholic University of Leuven, 2012.

References

1. https://staff.aist.go.jp/hori.y/en/puf/index.html. Physically Unclonable Function. Tokio (Japan): National Institute of Advanced Industrial Science and Technology, 2011.
2. https://helpiks.org/9-21094.html. Theoretical laws of the distribution of errors of quality parameters in the production of electronic means, 2014.
3. *Ahmed Ferozpuri.* Modeling delay-based PUFs with Machine Learning. Fairfaix (USA): GMU Electrical and Computer Engineering, 2015.
4. *Roel Maes.* Physically Unclonable Functions: Constructions, Properties and Applications. Leuven (Belgium): Catholic University of Leuven, 2012.

Поступила в редакцию 24.12.2019