Методика идентификации оборудования Интернета вещей по сектору деградировавшей флеш-памяти

С. С. Владимиров

28 января 2021 г.

Развитие сетевых технологий и, в особенности, Интернета вещей и сенсорных сетей, приводит к повсеместному использованию сетевых устройств. Они проникли практически во все области человеческой деятельности — производство, сельское хозяйство, образование, медицину и многие другие [1, 2, 3]. Такой уровень проникновения позволяет решать многие существующие в этих отраслях задачи, например, задачу автоматизации, но при этом ставит перед инженерами и разработчиками новые задачи, среди которых на первый план несомненно выходит информационная безопасность [1, 4, 5, 6].

Одним из основным вопросов, относящихся к информационной безопасности Интернета вещей, является задача однозначной идентификации устройства, которая должна обеспечить невозможность подмены этого устройства другим — ложным — с целью осуществления того или иного вида угрозы безопасности — сетевая атака, утечка или подмена информации и тому подобное [1, 5, 7]. В применении к задачам автоматизации эту проблему можно сформулировать иначе. Когда речь идёт об управляемом и управляющем устройствах, управляемое устройство должно однозначно определить, что получает команды именно от нужного управляющего устройства, и наоборот, управляющее устройство должно знать, что оно отсылает команды и получает данные именно от нужного управляемого устройства.

В настоящее время для идентификации сетевых устройств чаще всего используются идентификаторы, записываемые производителем в память сетевого устройства. Характерным примером таких идентификаторов служат широко используемые идентификаторы EUI-48 и EUI-64 — MAC-адреса сетевых устройств, зачастую используемые в качестве младшей части сетевого адреса IPv6; цифровые идентификаторы объекта DOI; а также предложенные китайскими компаниями системы CID (Communication Identifier) и Ecode (Entity Code) [8]. Однако, такие идентификаторы можно как изменить программным путём, так и подделать, прописав в память идентификатор другого устройства [9]. Другим вариантом идентификации является использование RFID-меток [7, 8]. Однако, они требуют использования специальных считывающих устройств и также не обеспечивают должного уровня безопасности [7]. Именно поэтому в последние годы проводятся исследования методик однозначной идентификации сетевых устройств и устройств Интернета вещей. В частности, предлагаются методы идентификации устройств по их физическим характеристикам или особенностям радиосигнала [10]. Другим перспективным вариантом является идентификация устройств по свойствам встроенного в них запоминающего устройства. Например, предлагается использовать встроенную в устройства флеш-память в качестве аппаратного генератора случайных чисел и источника уникальных идентификаторов устройства [11, 12]. Также, ранее был предложен метод использования слепка деградированной флеш-памяти в качестве идентификатора сетевого устройства [13]. В данной работе рассматривается вариант последнего метода идентификации, основанный на принудительной деградации участка флеш-памяти.

1 Принцип идентификации

Полупроводниковая флеш-память представляет собой массив элементарных ячеек памяти, размещенных на матрице проводников в виде двумерного (NOR-флеш) или трехмерного (NAND-флеш) массива. Каждая элементарная ячейка памяти может хранить от одного

до четырех бит информации (от двух до шестнадцати уровней заряда) в зависимости от типа ячейки. При операциях стирания и записи в ячейках памяти меняется хранимое значение уровня заряда. При этом в структуре ячеек памяти накапливаются необратимые изменения — чип деградирует, что постепенно приводит к появлению так называемых «бэд-блоков» — битовых ячеек памяти, которые не меняют свое состояние при операциях стирания и записи. Например, в флеш-памяти типа NOR исходным состоянием битовой ячейки является 1, а при записи информации в память нужные ячейки программируются нулями. Однако после большого числа перезаписей битовая ячейка может перестать возвращаться в исходное состояние и ее значение становится всегда равным 0 — возникает «бэд-блок» [14, 15]. В дальнейшем в статье также будут рассматриваться именно чипы NOR-флеш памяти, поскольку именно NOR флеш-память со свободным доступом используется в сетевых устройствах и их процессорах для хранения микропрограмм [15]. На рис. 1 приведен пример страницы деградировавшей NOR флеш-памяти в стравнении со страницей правильно работающей памяти.

Page of correctly working NOR-flash after erasing

	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
00	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
01	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
02	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
03	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
04	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
05	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
06	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
07	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
08	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
09	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
0A	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
0B	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
0C	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
0D	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
0E	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
0F	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF

Page of degraded NOR-flash after erasing

	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
00	FF	FF	FF	FF	FF	FE	FF	FF	3F	FF	FF	FF	EF	FF	FF	FF
01	FF	FF	FF	FF	FF	FF	FF	FF	DF	FF	FF	FF	FB	FF	DF	FF
02	FF	FF	FF	FF	FB	FB	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
03	FF	FF	FF	FF	7F	FF	F7	FF	EB	FF	FF	FF	FF	FF	FF	FF
04	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	EF	FF	FF	F7	FF
05	FF	FF	FF	FF	FF	FF	7F	FF	FF	FF	BF	FF	FF	FF	FF	FF
06	FF	FF	DB	FF	FF	FF	FF	FF	FF	FF	FB	FF	FD	FF	FF	FF
07	BF	FF	FF	FF	FF	FF	7F	FF	FF	BF	FF	FF	FF	FF	FF	FF
08	FF	FF	FF	BF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
09	FF	FF	FF	F4	FF	FF	FF	FF	FB	FF	FD	7F	FF	FD	FF	FF
0A	FB	FB	EF	FF	FF	FF	FF	FF	FF	FF	EF	FF	FF	FF	FD	FB
0B	FF	FE	FF	FF	FF	FF	FF	FF	FF	FF	FF	BF	FF	FF	FF	FF
0C	FF	F7	FF	FF	FF	FF	FF	FF	FF	FB	FF	FF	FF	FF	FF	FF
0D	FF	FF	FF	FF	FF	FF	FF	FF	F7	FF	FD	FB	FF	FF	FF	FF
0E	FF	FF	DF	FF	FF	FF	FF	BF	FF	FF	EF	FF	FF	EF	FF	FF
0F	DF	FF	FF	F7	DF	FF	FF	FF	FF	FF	FF	FF	FD	FF	FF	EF

Рис. 1: Сравнение страниц правильно работающей и деградировавшей NOR флеш-памяти

Утверждается, что каждый чип памяти и даже каждый участок массива ячеек памяти в одном чипе из-за особенностей производства является уникальным [11]. Это приводит к тому, что в каждом чипе флеш-памяти при деградации «бэд-блоки» возникают случай-

ным образом, следовательно, если взять два деградировавших чипа флеш-памяти, то рисунок «бэд-блоков» в них будет уникальным. Таким образом, этот рисунок «бэд-блоков» можно использовать в качестве уникального идентификатора, который теоретически невозможно повторить на другом чипе флеш-памяти [13].

С точки зрения архитектуры и порядка работы с памятью обычный чип NOR-флеш имеет иерархическую структуру, показанную на рис. 2. Весь массив памяти чипа разделен на крупные участки — блоки, каждый из которых поделен на секторы меньшего размера, в свою очередь состоящие из минимальных структурных участков — страниц. При этом минимальным участком для которого доступна операция стирания, является сектор, а операция программирования флеш-памяти производится постранично. Соответственно, чтобы изменить хотя бы один бит, хранящейся в чипе информации, необходимо использовать следующий алгоритм [11, 14, 15]:

1. Чтение всей информации из соответствующего сектора.
2. Изменение необходимой информации в считанном массиве данных.
3. Стирание всего считанного сектора.
4. Постраничная запись информации в стертый сектор.

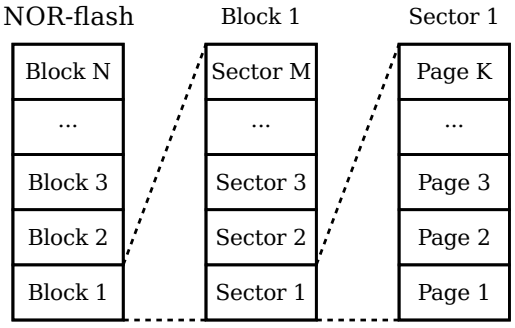

Рис. 2: Структура NOR флеш-памяти

Поэтому в NOR-флеш памяти минимальным участком, который может быть подвержен деградации, является именно сектор памяти. Таким образом, поскольку во всех современных сетевых устройствах уже присутствует чип флеш-памяти для хранения микропрограмм, в качестве уникального идентификатора можно использовать один сектор памяти этого чипа, осуществив процедуру его принудительной деградации. Однако, такой способ может быть использован только в том случае, если деградация этого сектора памяти, не затронет соседние сектора памяти. В противном случае можно добавить в сетевое устройство отдельный чип флеш-памяти, предназначенный только для идентификации этого устройства.

2 Методика и результаты исследования

Для тестирования метода идентификации были выбраны чипы NOR флеш-памяти емкостью 16 Мбит (2 Мбайт) в корпусе SO-8 и последовательным интерфейсом SPI. Именно такие микросхемы используются в большинстве сетевых устройств. Проверка производилась для чипов EN25F16 производства Eon Silicon Solution, Inc. и чипов W25Q16V производства Winbond Electronics Corporation. Эти чипы имеют одинаковую структуру памяти, состоящую из $N = 32$ блоков по 64 кбайт каждый, которые в свою очередь содержат $M = 16$ секторов по 4 кбайт (всего 512 секторов на чип), состоящих из $K = 16$ страниц по 256 байт (всего 8192 страницы на чип). Выбранные чипы используют стандартную систему команд (инструкций).

Экспериментальная установка была построена на основе микрокомпьютера Raspberry Pi B, содержащего встроенный SPI интерфейс, и подключенного к его разъемам GPIO программатора SPI, построенного по стандартной схеме. Схема экспериментальной установки показана на рис. 3.

Для работы с чипом памяти было написано программное обеспечение на языке Python с использованием модуля py-spidev, определяющего функции для работы с драйвером spidev ядра ОС Linux, который обеспечивает низкоуровневые механизмы по работе с интер-

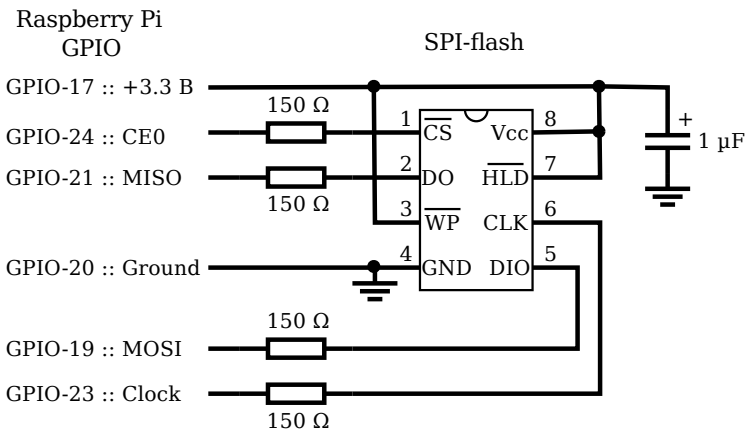

Рис. 3: Схема экспериментальной установки

фейсом SPI. Программное обеспечение выполняет следующие функции:

1. Циклическая перезапись одного сектора чипа флеш-памяти до устойчивого появления «бэд-блоков».

2. Сохранение образа (рисунка «бэд-блоков») деградировавшего сектора чипа флеш-памяти.

3. Сравнение сектора подключенного к экспериментальной установке чипа флеш-памяти с сохраненными образами и идентификация чипа по методу максимального правдоподобия.

Циклическая перезапись осуществляется путем стирания сектора флеш-памяти соответствующей командой и последующей постраничной записи сектора массивом нулей, что обеспечивает равновероятность потенциальной деградации элементарных ячеек памяти сектора.

В результате проведенного эксперимента были успешно деградированы сектора каждого исследованного чипа памяти. При этом были определены два основных свойства:

1. Рисунок «бэд-блоков» деградировавших 4 кбайт секторов каждого исследованного чипа действительно отличается, что поз-

волило провести идентификацию чипов в рамках исследованной выборки. Число «бэд-блоков», совпадающих при сравнении сектора одного чипа памяти с секторами других чипов памяти, оказалось на два порядка меньше числа «бэд-блоков», совпадающих при сравнении с ранее записанным в память рисунком «бэд-блоков» этого сектора.

2. Некоторые искаженные биты («бэд-блоки») при стирании могут иногда возвращаться в исходное состояние. Таким образом, для получения полного рисунка «бэд-блоков», необходимо провести несколько стираний сектора подряд, накапливая позиции «бэд-блоков». После этого получившийся образ сектора можно использовать для идентификации чипа.

Возможность принудительного деградирования одного сектора флеш-памяти без деградации остальных секторов в рамках проведенного эксперимента подтвердить не удалось, поскольку один из чипов по невыясненной причине деградировал полностью. Данное предположение будет проверяться в дальнейших экспериментах.

Также необходимо отметить, что экспериментальная установка на основе микрокомпьютера класса Raspberry Pi и программного обеспечения на интерпретируемом языке Python при всей простоте работы с ней обеспечивает скорость деградации чипа, недостаточную для проведения массовых экспериментов на больших выборках. В связи с этим в дальнейших экспериментах предполагается использовать и сравнить несколько вариантов экспериментальной установки:

1. Экстенсивно расширенную установку из большого количества параллельно работающих микрокомпьютеров.

2. Разрабатываемую установку на базе параллельно включенных устройств USB–SPI, управляющего компьютера и программного обеспечения на языке C.

3. Установку, в которой механизм деградации сектора памяти вынесен в отдельное устройство на основе быстрых ПЛИС.

3 Процедура идентификации

Исходя из полученных результатов была сформулирована процедура идентификации сетевого устройства с использованием деградировавшего чипа флеш-памяти.

1. Принудительное деградирование чипа флеш-памяти путем циклической перезаписи одного из его секторов (в эксперименте использовался первый сектор) до устойчивого появления «бэд-блоков».

2. Формирование уникального массива-идентификатора $\mathbf{S}_{\mathrm{ID}}(c)$ чипа памяти c. Для этого изначально формируется нулевой массив $\mathbf{S}_{\mathrm{ID}}(c)$, соответствующий по размеру сектору чипа памяти. Затем деградированный сектор стирается десять раз подряд. После каждого i стирания содержимое сектора считывается в массив $\mathbf{B}_i$. Далее осуществляется поэлементная конъюнкция массивов $\mathbf{B}_i$ с записью результата в $\mathbf{S}_{\mathrm{ID}}(c)$. В итоге, после десяти стираний массив $\mathbf{S}_{\mathrm{ID}}(c)$ содержит уникальный идентификатор чипа памяти c. Эту операцию можно описать формулой

$$\mathbf{S}_{\mathrm{ID}}(c) = \prod_{i=1}^{10} \mathbf{B}_i,$$

   где символ произведения соответствует операции поэлементного логического умножения (конъюнкции).

3. Массив-идентификатор $\mathbf{S}_{\mathrm{ID}}(c)$ каждого обработанного таким образом чипа памяти c записывается в хранилище идентификаторов.

4. Для идентификации сетевого устройства, содержащего идентификационный чип памяти, производится стирание выбранного сектора чипа памяти, считывание содержимого сектора после стирания и сравнение результата с каждым идентификатором $\mathbf{S}_{\mathrm{ID}}$ из хранилища идентификаторов. Соответствие идентификатора определяется по мажоритарному принципу.

Простейшим вариантом использования этой методики является применение деградированных чипов флеш-памяти в качестве аппаратных идентификаторов, содержимое которых считывается напрямую специальным считывателем по SPI интерфейсу. Такой считыватель выполняет всю процедуру идентификации: стирание идентифицирующего сектора, считывание уникального массива-идентификатора $\mathbf{S}_{\mathrm{ID}}(c)$, запрос в хранилище идентификаторов (либо поиск во встроенном хранилище при его малом размере) и вывод результата поиска.

Надежность идентификации сетевого устройства, содержащего идентификационный чип памяти, напрямую зависит от размера выбранного в качестве идентификатора блока. Чем больше размер блока, тем выше надежность идентификации. Однако, с увеличением размера идентификатора растут время его обработки и сравнения с содержимым хранилища идентификаторов. Таким образом, для простой идентификации можно использовать укороченный блок-идентификатор, например, одну или две страницы памяти. Будем обозначать такой укороченный идентификатор $\mathbf{P}_{\mathrm{ID}}(c)$. Для идентификации при важных задачах стоит использовать блок-идентификатор $\mathbf{S}_{\mathrm{ID}}(c)$ большего размера, например, целый сектор или несколько секторов. При критичных задачах можно применять в качестве идентификатора рисунок «бэд-блоков» для всего чипа памяти.

Этот принцип может быть применен для идентификации в рамках сеанса передачи данных между двумя сетевыми устройствами. Рассмотрим случай, при котором проверка идентификаторов производится в отдельном сетевом устройстве — хранилище идентификаторов — которое хранит заведомо верные образы полных идентификаторов сетевых устройств $\mathbf{S}_{\mathrm{ID}}$ и сопоставляет их с использующимися в данной сети передачи данных сетевыми идентификаторами (сетевыми адресами) N_{ID}, формируя таблицу идентификации. Схема сети для рассматриваемого примера показана на рис. 4(1). На рисунке приведены хранилище идентификаторов IS и два сетевых устройства D1 и D2, имеющих идентификаторы $\mathbf{S}_{\mathrm{ID}}(\mathrm{D1}) \to N_{\mathrm{ID}}(\mathrm{D1})$ и $\mathbf{S}_{\mathrm{ID}}(\mathrm{D2}) \to N_{\mathrm{ID}}(\mathrm{D2})$, соответственно.

Изначально в хранилище идентификаторов IS должны быть внесены образы полных идентификаторов $\mathbf{S}_{\mathrm{ID}}$ всех сетевых устройств

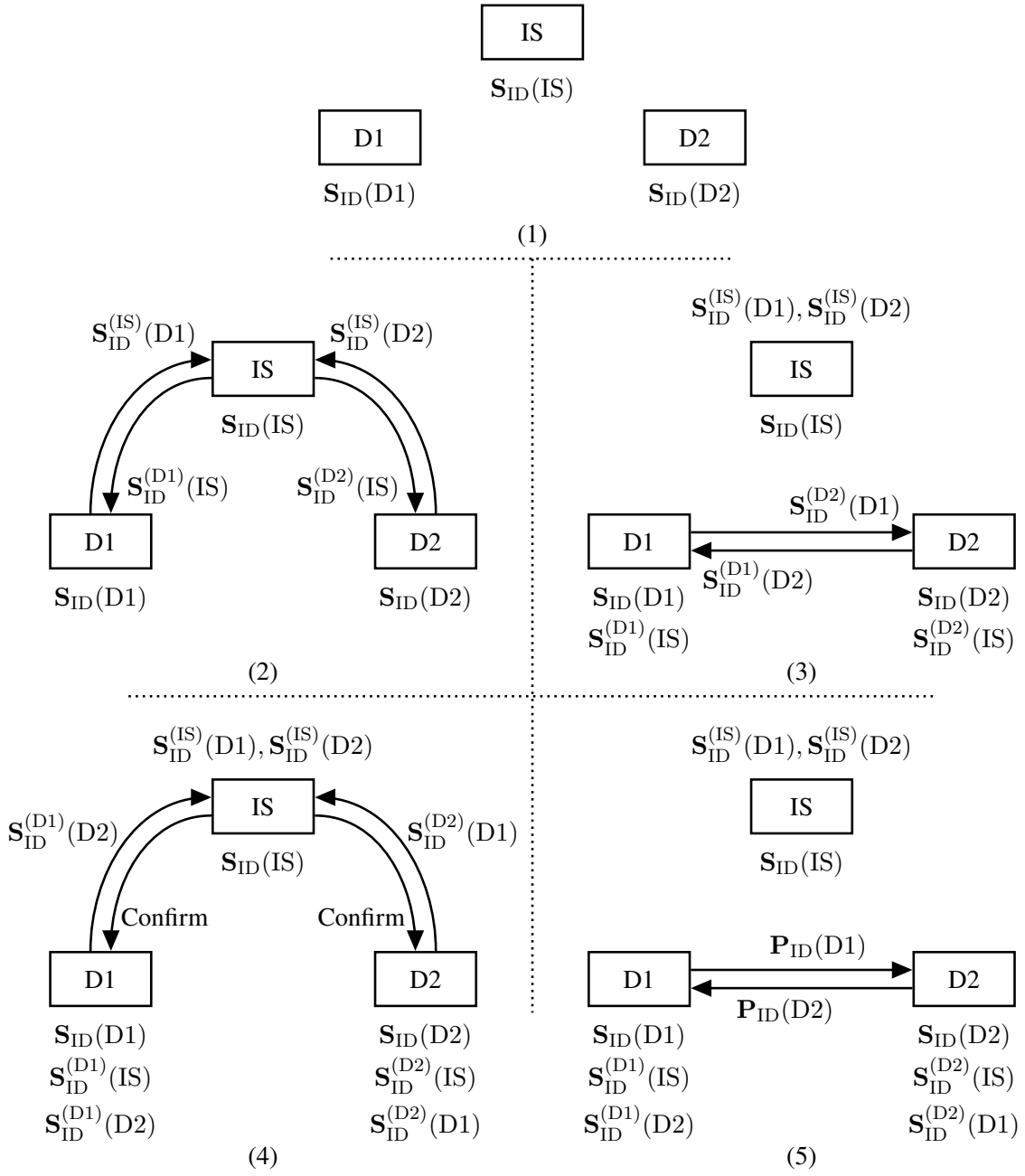

Рис. 4: Упрощенная схема идентификации сетевых устройств

в данной сети. Обозначим образы идентификаторов для устройств D1 и D2 как $\mathbf{S}_{\mathrm{ID}}^{(\mathrm{IS})}(\mathrm{D1})$ и $\mathbf{S}_{\mathrm{ID}}^{(\mathrm{IS})}(\mathrm{D2})$. Эти образы могут быть внесе-

ны в хранилище администратором сети либо записаны автоматически при прямом подключении устройства к IS. Уникальный полный идентификатор самого хранилища $\mathbf{S}_{\mathrm{ID}}(\mathrm{IS})$ прописывается в память каждого устройства, чтобы при обращении к нему устройство могло быть уверено, что работает именно к IS. Эта процедура показана на рис. 4(2).

При установлении соединения, каждое устройство инициализирует чип памяти, использующийся для идентификации, стирая блок-идентификатор. Затем устройства считывают свои полные идентификаторы $\mathbf{S}_{\mathrm{ID}}$ и в процессе установления соединения обмениваются ими, как показано на рис. 4(3), чтобы подтвердить подлинность. После получения полного идентификатора собеседника, устройство D2 отправляет запрос в хранилище IS, в котором указывает сетевой $N_{\mathrm{ID}}(\mathrm{D1})$ и полный $\mathbf{S}_{\mathrm{ID}}(\mathrm{D1})$ идентификаторы собеседника. IS сравнивает полученный идентификатор $\mathbf{S}_{\mathrm{ID}}(\mathrm{D1})$ с хранящимся в его памяти образом $\mathbf{S}_{\mathrm{ID}}^{(\mathrm{IS})}(\mathrm{D1})$ по мажоритарному принципу. Если проверяемый идентификатор и сохраненный контрольный образ коррелируют, то IS отправляет D2 подтверждение, в котором указывает свой уникальный полный идентификатор $\mathbf{S}_{\mathrm{ID}}(\mathrm{IS})$, который служит доказательством того, что подтверждение получено от валидного источника. Получив подтверждение, D2 записывает в свою память полный идентификатор D1 в виде образа $\mathbf{S}_{\mathrm{ID}}^{(\mathrm{D2})}(\mathrm{D1})$. В дальнейшем, этот образ используется для динамической идентификации устройств в процессе работы. Аналогичные действия производит устройство D1 в отношении идентификатора $\mathbf{S}_{\mathrm{ID}}(\mathrm{D2})$. Эта процедура показана на рис. 4(4).

Далее, в процессе передачи данных можно использовать укороченный идентификатор $\mathbf{P}_{\mathrm{ID}}$ для периодического подтверждения подлинности, как показано на рис. 4(5). При этом можно использовать как статический укороченный идентификатор, например, только первую страницу полного идентификатора, так и различные варианты динамической идентификации, к примеру чередуя укороченные идентификаторы по тому или иному алгоритму либо используя циклический сдвиг укороченного идентификатора в рамках полного иденти-

фикатора. В этом случае можно использовать как синхронную динамическую идентификацию, когда каждое устройство знает, какую часть полного идентификатора $\mathbf{S}_{\mathrm{ID}}$ оно должна получить в качестве $\mathbf{P}_{\mathrm{ID}}$, так и асинхронную динамическую идентификацию, при которой соответствие $\mathbf{P}_{\mathrm{ID}}$ и $\mathbf{S}_{\mathrm{ID}}$ определяется по выбросу взаимнорреляционной функции

$$\mathrm{CCF}_i(\mathbf{P}_{\mathrm{ID}}, \mathbf{S}_{\mathrm{ID}}) = \sum_j \mathbf{P}_{\mathrm{ID}\,j} \oplus \mathbf{S}_{\mathrm{ID}\,i+j}.$$

По окончании сеанса связи, сохраненные образы полных идентификаторов собеседников можно либо стирать, либо сохранять и использовать в дальнейшем для упрощенной идентификации, не требующей обращения к хранилищу идентификаторов.

Необходимо отметить, что частое считывание содержимого деградированного сектора, сопряженное с его стиранием, может привести к его дальнейшей деградации и появлению новых «бэд-блоков». Следовательно, после успешной идентификации чипа по мажоритарному принципу, вновь появившиеся «бэд-блоки» следует добавлять к сохраненному в хранилище образу идентификатора.

Таким образом, проведенный экперимент подтвердил принципиальную возможность использования участка деградированной флеш-памяти в качестве уникального идентификатора. Также была сформулирована процедура идентификации сетевого устройства с использованием сектора деградировавшей флеш-памяти и предложены варианты ее применения. Авторы планируют продолжать исследования в этой области и предложенная процедура не является окончательной. Планируется провести исследования на надежность идентификации с определением вероятностей правильной и неправильной идентификации устройств. В зависимости от результатов будущих исследований, процедура идентификации может быть усовершенствована в сторону упрощения или, наоборот, усложнения, исходя из критериев максимальных надежности и быстродействия.

Список литературы

[1] Rose K, Eldridge S, Chapin L (2015) The Internet of Things: An Overview. Understanding the Issues and Challenges of a More Connected World. https://www.internetsociety.org/sites/default/files/ISOC-IoT-Overview-20151014_0.pdf. Accessed 1 May 2017

[2] Greengard S (2015) The Internet of Things. Mit Press

[3] Vermesan O, Friess P (eds) (2013) Internet of Things: Converging Technologies for Smart Environments and Integrated Ecosystems. River Publishers, Aalborg

[4] Russell B, Van Duren D (2016) Practical Internet of Things Security. Packt Publishing

[5] Dhanjani N (2015) Abusing the Internet of Things. Blackouts, Freakouts, and Stakeouts. O'Reilly Media

[6] Fei Hu (ed) (2016) Security and Privacy in Internet of Things (IoTs): Models, Algorithms, and Implementations. CRC Press, Boca Raton

[7] Leloglu E (2017) A Review of Security Concerns in Internet of Things. Journal of Computer and Communications. 5:121–136

[8] Soldatos J, Yuming G (eds) (2014) Internet of Things. EU-China Joint White Paper on Internet-of-Things Identification. European Research Cluster on the Internet of Things

[9] Hegde A (2016) MAC Spoofing Detection and Prevention. International Journal of Advanced Research in Computer and Communication Engineering. 5(1):229–232

[10] DeJean G, Kirovski D (2007) RF-DNA: Radio-Frequency Certificates of Authenticity. Lecture Notes in Computer Science. 4727:346–363

[11] Wang Y et al (2012) Flash Memory for Ubiquitous Hardware Security Functions: True Random Number Generation and Device Fingerprints. In: Proceedings of the 2012 IEEE Symposium on Security and Privacy, IEEE Computer Society, Washington, DC, 20–25 May 2012. doi:10.1109/SP.2012.12

[12] Jia S, Xia L, Wang Z, Lin J, Zhang G, Ji Y (2015) Extracting Robust Keys from NAND Flash Physical Unclonable Functions. In: Lopez J, Mitchell C (eds) Information Security. ISC 2015. Lecture Notes in Computer Science, vol 9290. Springer, Cham

[13] Jakobsson M, Johansson K-A (2010) Unspoofable Device Identity Using NAND Flash Memory. http://www.securityweek.com/unspoofable-device-identity-using-nand-flash-memory. Accessed 1 May 2017

[14] Bez R, Camerlenghi E, Modelli A, Visconti A (2003) Introduction to Flash Memory. Proceedings of the IEEE. 91(4): 489–502. doi:10.1109/JPROC.2003.811702

[15] Tal A (2002) Two Flash Technologies Compared: NOR vs NAND. 91-SR-012–04–8 L REV. 1.0. M-Systems Flash Disk Pioneers. Newark