**Описание протокола взаимодействия с сервисом ATECC508A/608A.**

Формат данных JSON, протокол обмена UDP, порт 7777. IP адрес: 127.0.0.0. Для тестирования открыт любой адрес.

Все запросы/ответы от **сервиса** приходят в JSON формате и имеют вид:

{  
 **"channel"**: **1**, *// номер канала 0,1 или 2.*

**"id"**:**1**, *// ID запроса, байт 0-255*

**"command"**:**"open\_channel"**, *// команда в канал*}

Если в процессе выполнения команды возникает ошибка, то формируется ответ с кодом ошибки.

{

**"channel":1**,

**"id":6**,

**"command":"create\_key",**

**"error":244**

}

Обязательные параметры, которые всегда присутствуют в запросе и ответе это номер канала (**channel**), ID запроса (**id**) и команда (**command**).

Параметр **channel** может принимать только три значения

**"channel"**: **0** – программный канал

**"channel"**: **1** – 1-я микросхема ATECC508A (Hardware channel N1)

**"channel"**: **2** – 2-я микросхема ATECC508A (Hardware channel N2)

Параметр **id** нужен для уникальности запроса и ответа в текущий момент времени. Ответ приходит с тем же ID что и запрос. Размерность ID байт.

Параметр **command** имеет текстовое значение и может принимать следующие значения:

**"open\_channel"** – открыть программный или аппаратный канал;

**"close\_channel"** – закрыть программный или аппаратный канал;

(команды аппаратного канала)

**"get\_sn"** – команда получения 72 битного серийного номера микросхемы ATECC508A;

**"get\_locks"** – команда состояния защиты микросхемы;

**"get\_config"** – команда чтения конфигурации микросхемы;

**"create\_key"** – генерация приватного ключа ECC 256бит в защищенной памяти. Приватный ключ никогда не покидает защищенную память и нет способов для его извлечения;

**"get\_public\_key"** – формирование публичного ключа ECC 256бит на основе приватного ключа сохраненного в защищенной памяти;

**"get\_rng"** – формирование аппаратного случайного числа нужной длины;

**"sign"** – формирование цифровой подписи ECDSA

**"verify"** – проверка цифровой подписи по публичному ключу;

**"verify2"** - проверка цифровой подписи на основе приватного ключа сохраненного в защищенной памяти;

**"ecdh"** – Диффи-Хеллман на основе приватного ключа сохраненного в защищенной памяти;

**"ecdhe"** – Диффи-Хеллман на основе эфемерного (TempKey) ключа, временного сгенеренного ключа и хранящегося в ОЗУ ATECC508A;

**"create\_cert"** – генерация нового приватного ключа ECC 256бит, формирование самоподписанного сертификата и сохранение его в защищенной памяти;

**"get\_cert"** - считывание самоподписанного сертификата из защищенной памяти;

Команда открыть программный или аппаратный канал

Пример обмена открытия аппаратного канала N1.

Запрос: *{"channel":1,"id":1,"command":"open\_channel"}*

Ответ: *{"channel":1,"id":1,"command":"open\_channel","response":"00"}*

**"response"** – шестнадцятиричный код (результат) открытия канала, должен быть всегда «00» иначе ошибка.

Команда получения 72 битного серийного номера микросхемы ATECC508A

Запрос: {"channel":1,"id":1,"command":"get\_sn"}

Ответ:{"channel":1,"id":1,"command":"get\_sn","response":"0123221194039F31EE"}

**"response"** – 72 бит (9 байт) серийного номера микросхемы ATECC508A канала 1.

Команда получения состояния защиты микросхемы

Запрос: {"channel":1,"id":1,"command":"get\_locks"}

Ответ:{"channel":1,"id":1,"command":"get\_locks","response":"010100"}

**"response"** – должен быть "010100" (Lock Config, Lock OTP, Unlock Keys)

Команда генерация приватного ключа ECC 256бит

Запрос: {"channel":1,"id":1,"command":"create\_key","key\_id":1}

"key\_id": 1 – 7 – номера ID ключа ATECC508A. "key\_id":0 – используется для подписи сертификата.

Ответ:{"channel":1,"id":10,"command":"create\_key",

"response":"04765BF4F078DF7E0A9A4801F5C4E8863120B5761951B9B8945C4BBC0ADAA880321A15AF745BE869C618682FCCD857CCD49C14485D7F0EE6A8C485800DA2F4F20F"}

**"response"** – 65 байт. Публичный ECC 256бит ключ в некомпрессированном формате.

Команда чтения приватного ключа ECC 256бит

Запрос: {"channel":1,"id":1,"command":"get\_public\_key","key\_id":0}

Ответ:{"channel":1,"id":1,"command":"get\_public\_key",

"response":"04765BF4F078DF7E0A9A4801F5C4E8863120B5761951B9B8945C4BBC0ADAA880321A15AF745BE869C618682FCCD857CCD49C14485D7F0EE6A8C485800DA2F4F20F"}

**"response"** – 65 байт. Публичный ECC 256бит ключ в бинарном формате.

Первый байт всегда 0x04, за которым следуют 32 байта X, затем 32 байта Y.

Команда генерации случайного числа.

Запрос: {"channel":1,"id":1,"command":"get\_rng","request":"8000"}

Request – длина запрашиваемого случайного числа, в примере 128 байт

Ответ:{"channel":1,"id":1,"command":"get\_rng",

"response":"490A11FF6997AB8CAD6BFE9E9D6966FC4632F36C9E19B4BF166B3AE7B5BF7911441B7640847EAD6F3B7C211424BEFD096813E33FFA2CAAB21D2ADFBEE8B5C43278285D4CBA5C6C96EF31F9832DCDCC61B6357070261D66918EC34E951FFD4B8DA3726716B815454C2F8F78ED8F342A63B32E63C3E550A25880336E344CE6AA38"}

**"response"** –сгенеренное случайное число

Команда формирования цифровой подписи ECDSA

Запрос: {"channel":1,"id":1,"command":"sign","key\_id":0,

"hash":"0102030405060708090A0B0C0D0E0F101112131415161718191A1B1C1D1E1F20"}

где "key\_id": (0- 7) ID нужного ключа;

"hash" – SHA256 (32 байта) хеш данных

Ответ:

{"channel":1,"id":13,"command":"sign","response":"B686A4B6C50C7AA3C9DCAAD05404343FF386049636B7C3434A9FA64CF8D46DB8C82DB2D18833A2AA62091EEB4608C7AD05A3C1F3CE249381572A7FEBEE541287"}

**"response" –** 64 байта цифровой подписи

Команда проверка цифровой подписи по публичному ключу, который передается в запросе.

Запрос:

{

"channel":1,

"id":14,

"command":"verify",

"hash":"0102030405060708090A0B0C0D0E0F101112131415161718191A1B1C1D1E1F20",

"sign":"F9E90963E9463DBD98C52798B421988610CAA28E1A63F28FC088D050364D6B79166C15618AC7388383C356BB46687E6720085CDF6420C5D6E9043BA2A313CA41",

"public\_key":"04765BF4F078DF7E0A9A4801F5C4E8863120B5761951B9B8945C4BBC0ADAA880321A15AF745BE869C618682FCCD857CCD49C14485D7F0EE6A8C485800DA2F4F20F"}

Ответ: {"channel":1,"id":14,"command":"verify","response":"01"}

**"response" -** 1 байт результата проверки "01" – true "00" –false

Команда проверка цифровой подписи по ID ключа. Публичный ключ не передается, он берется из модуля.

Запрос:

{

"channel":1,

"id":1,

"command":"verify2",

"key\_id":0,

"hash":"0102030405060708090A0B0C0D0E0F101112131415161718191A1B1C1D1E1F20",

"sign":"0D325DF53672BAE9F8ACDA2DCE856DC5A0B8152FC4E90EE9B1E828230CF047012D9148E0600B78C822AD0EFB2BD35804D3D1371408B673E4A008B8E324F4B1D1"}

Ответ: {"channel":1,"id":1,"command":"verify2","response":"01"}

Команда протокола Диффи-Хеллмана на основе статического приватного ключа сохраненного в защищенной памяти (KEY ID). В запросе передается 65 байт открытого ключа абонента.

Запрос:

{

"channel":1,

"id":18,

"command":"ecdh",

"key\_id":0,

"public\_key":"046FD311BC1C334D752CEAC420F9A9DAE56DD5E3D643A2DA4057CAE38D1DF6F50E91714C5EC9581C02784AC7D6C3F07E68F7E924472E87878F9C0B130F92A9E86E"}

В ответе возвращается открытый ключ (65байт) на основе сохраненного приватного ключа и общий секрет длиной 32 байта.

Ответ: {"channel":1,"id":18,"command":"ecdh","response":"0416CE266419583924B423B8334BF50BD3FB64F39D9F6B43CCAD0E3164A2F9F7E801762D2505F1DD08DEE5C769C401FEFBF8E6176F5804BD3C09996DEA8E406A262F340B9F13334714A2061334E5469C2277D6B68B892E2B3CA70EBEDF18CE37B5"}

Команда протокола Диффи-Хеллмана на основе эфемерного (TempKey) приватного ключа.

В запросе передается 65 байт открытого ключа абонента.

Запрос:

{"channel":1,"id":19,"command":"ecdhe","public\_key":"046FD311BC1C334D752CEAC420F9A9DAE56DD5E3D643A2DA4057CAE38D1DF6F50E91714C5EC9581C02784AC7D6C3F07E68F7E924472E87878F9C0B130F92A9E86E"}

Ответ:

{"channel":1,"id":19,"command":"ecdhe","response":"041F2485D4B75393381F2103C941377AA411E56A46D751878ED171B4521C5E5B160E41820CF409DDED2BA5362C5CFC20B71F11F806423A5817AC95255EE9B8E71ED8E76DEF67735FDD2AD539B0513605B623CDF39E047E403B9AED39101F6C2224"}

Команда генерации нового сертификата и сохранение его в памяти ATECC508A.

KEY ID не передается и по умолчанию он равен "key\_id":0

Запрос:

{"channel":1,"id":20,"command":"create\_cert"}

Ответ:

{"channel":1,"id":20,"command":"create\_cert","response}

Команда чтения сертификата из защищенной памяти.

Запрос:

{"channel":1,"id":21,"command":"get\_cert"}

Ответ:

{"channel":1,"id":21,"command":"get\_cert","response}