Chipsmall Limited consists of a professional team with an average of over 10 year of expertise in the distribution of electronic components. Based in Hongkong, we have already established firm and mutual-benefit business relationships with customers from,Europe,America and south Asia,supplying obsolete and hard-to-find components to meet their specific needs.

With the principle of “Quality Parts,Customers Priority,Honest Operation,and Considerate Service”,our business mainly focus on the distribution of electronic components. Line cards we deal with include Microchip,ALPS,ROHM,Xilinx,Pulse,ON,Everlight and Freescale. Main products comprise IC,Modules,Potentiometer,IC Socket,Relay,Connector.Our parts cover such applications as commercial,industrial, and automotives areas.

We are looking forward to setting up business relationship with you and hope to provide you with the best service and solution. Let us make a better world for our industry!

Contact us

Tel: +86-755-8981 8866 Fax: +86-755-8427 6832
Email & Skype: info@chipsmall.com Web: www.chipsmall.com
Address: A1208, Overseas Decoration Building, #122 Zhenhua RD., Futian, Shenzhen, China

ATSHA204A

ATSHA204A Microchip CryptoAuthentication Data Sheet

Features

- Crypto Element with Protected Hardware-Based Key Storage
- Secure Symmetric Authentication Device Host and Client Operations
- Superior SHA-256 Hash Algorithm with Message Authentication Code (MAC) and Hash-Based Message Authentication Code (HMAC) Options
- Best-in-class, 256-bit Key Length; Storage for Up to 16 Keys
- Guaranteed Unique 72-bit Serial Number
- Internal, High-quality Random Number Generator (RNG)
- 4.5 kb EEPROM for Keys and Data
- 512 bit OTP (One Time Programmable) Bits for Fixed Information
- Multiple I/O Options
  - UART-compatible High-Speed, Single-Wire Interface
  - 1 MHz I^2C Interface
- 2.0V to 5.5V Supply Voltage Range
- 1.8V to 5.5V Communications Voltage Range
- <150 nA Sleep Current
- Secure Download and Boot
  - Ecosystem Control
  - Message Security
  - Anti-Cloning
- 8-lead SOIC, 8-lead TSSOP (2), 3-lead SOT23, 8-pad UDFN and 3-lead CONTACT Packages

Applications

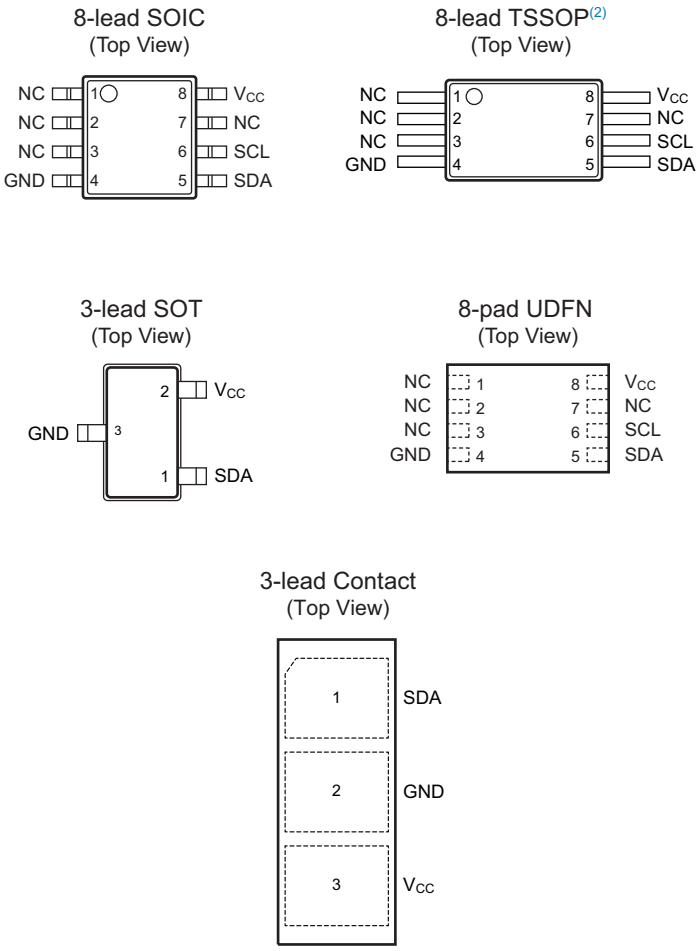
- Secure Download and Boot
- Ecosystem Control
- Anti-cloning
- Message Security

Package Types

Table 1. Pin Configuration

Pin Name	Function
NC	No Connect
GND	Ground
SDA	Serial Data
SCL	Serial Clock Input
V_{CC}	Power Supply

Figure 1. Pinouts(1)



Note:

1. Drawings are not to scale.
2. Not recommended for new design.

Table of Contents

1. Introduction

The following sections introduce the features and functions of the Microchip ATSHA204A crypto element device.

1.1 Applications

The ATSHA204A is a member of the Microchip CryptoAuthentication™ family of high-security hardware authentication devices. It has a flexible command set that allows use in many applications, including the following:

- **Anti-Counterfeiting**
  Validates that a removable, replaceable, or consumable client is authentic. Example of clients could be printer ink tanks, electronic daughter cards, medical disposables, or spare parts. The device can also be used to validate (authenticate) a software/firmware module or memory storage element.
- **Protecting Firmware or Media**
  Validates code that is stored in flash memory at boot time to prevent unauthorized modifications (this is also known as secure boot), encrypts downloaded media files and uniquely encrypts code images to be usable on a single system only.
- **Exchanging Session Keys**
  Securely and easily exchanges stream encryption keys for use by an encryption/decryption engine in the system microprocessor to manage a confidential communications channel, an encrypted download and similar items.
- **Storing Data Securely**
  Stores secret keys for use by crypto accelerators in standard microprocessors. It can also be used to store small quantities of data necessary for configuration, calibration, ePurse value, consumption data, or other secrets. Programmable protection up through encrypted/authenticated reads and writes.
- **Checking User Password**
  Validates user-entered passwords without letting the expected value become known, mapping simple passwords to complex ones and securely exchanging password values with remote systems.

1.2 Device Features

The ATSHA204A device includes an Electrically Erasable Programmable Read-Only Memory (EEPROM) array that can be used for key storage, miscellaneous read/write data, read-only, secret data, consumption logging and security configuration. Access to the various sections of memory can be restricted in a variety of ways and the configuration can then be locked to prevent changes. See Section **EEPROM Organization** for details.

The ATSHA204A features a wide array of defense mechanisms specifically designed to prevent physical attacks on the device itself or logical attacks on the data transmitted between the device and the system see Section **Security Features** for more details. Hardware restrictions on the way keys are used or generated provide further defense against certain styles of attack.

Access to the device is made through a standard I^2C interface at speeds of up to 1 Mb/s. see Section **I^2C Interface** for details. It is compatible with I^2C interface specifications. The device also supports a Single-Wire Interface (SWI) that can reduce the number of GPIOs required on the system processor and/or reduce the number of pins on connectors. See Section **Single-Wire Interface** for more details.

Using the Single-Wire Interface, multiple ATSHA204A devices can share the same bus, which saves processor GPIO usage in systems with multiple clients such as different color ink tanks or multiple spare parts, as examples. See Section **Sharing the Interface** and Section **`Pause` Command** for details on how this is implemented.

Each ATSHA204A ships with a guaranteed unique 9-byte (72-bit) serial number. Using the cryptographic protocols supported by the device, a Host system or remote server can prove that the serial number is authentic and is not a copy. Serial numbers are often stored in a standard Serial EEPROM, which can be easily copied with no way for the Host to know if the serial number is authentic or if it is a clone. The entire serial number must be utilized to guarantee uniqueness.

The ATSHA204A can generate high-quality random numbers and employ them for any purpose, including as part of the crypto protocols of this device. Because each 32-byte (256-bit) random number is not dependent on past numbers generated on this or any other device, their inclusion in the protocol calculation ensures that replay attacks (for instance. re-transmitting a previously successful transaction) always fail. See Section **Random Number Generator (RNG)** and Section **`Random` Command**.

System integration is made easy by a wide supply voltage range (of 2.0V through 5.5V) and an ultra-low sleep current (of <150 nA). Complete DC parameters are found in Section **Electrical Characteristics**, which describes multiple package options, including a tiny UDFN package with a footprint of only 2.0 mm x 3.0 mm. See Section **Package Drawings** for more details and ordering codes.

See Section **Compatibility** for information regarding compatibility with the Microchip ATSHA204.

1.3 Cryptographic Operation

The ATSHA204A supports a standard challenge-response protocol to simplify programming. In its most basic installation, the Host system sends a challenge (for example a number) to the device in the Client, which combines that challenge with a secret key by using the `Message Authentication Code (MAC)` command from the system, as described in Section **`MAC` Command** and sends that response back to the system. The device uses a cryptographic hash algorithm to make that combination (which is also known as a digest). The use of a hash algorithm prevents an observer on the bus from deriving the value of the secret key, while allowing the recipient to verify that the response is correct by performing the same calculation combining the challenge with the secret to create a digest using a stored copy of the secret.

This basic operation can be expanded in many ways because of the flexible command set of the ATSHA204A. By using the `GenDig` command (Section **`GenDig` Command**), the values in other slots can be included in the response digest, which provides an effective way of proving that a data read really did come from the device, as opposed to being inserted by a man-in-the-middle attacker. This same command can be used to combine two keys with the challenge, which is useful when there are multiple layers of authentication to be performed.

The `DeriveKey` command (Section **`DeriveKey` Command**) implements a key rolling scheme. Depending upon the command mode parameter, the resulting operation can be similar to that implemented in a remote-controlled garage door opener, for example. Each time the key is used, the current value of the key is cryptographically combined with a value specific to that system and that result then forms the key for the next cryptographic operation. Even if an attacker obtains the value of one key, that key disappears forever with the next use.

`DeriveKey` can also be used to generate new random keys that might be valid only for a particular Host ID, for a particular time period, or for some other restricted condition. Each generated key is different from

any other key ever generated on any device. By “activating” a Host-Client pair in the field in this manner, a clone of a single Client can not work on any other Host.

In a Host-Client configuration where the Host (for example a mobile phone) needs to verify a Client (for instance an OEM battery), there is a need to store the secret in the Host in order to validate the response from the Client. The `CheckMac` command (Section **`CheckMac` Command**) allows the Host device to securely store the Client’s secret and hide the correct response value from the pins, returning only a yes/no answer to the system.

Where a user-entered password is required, the `CheckMac` command also provides a way to both verify the password without exposing it on the communications bus and map the password to a stored value that can have much higher entropy. See Section **Password Checking** for details.

Finally, the hash combination (for example digest) of a challenge and secret key can be kept on the device and XORed with the contents of a slot to implement an encrypted read (Section **`Read` Command**), or it can be XORed with encrypted input data to implement an encrypted write (Section **`Write` Command**).

Each of these operations can be protected against replay attacks by including a random nonce (Section **`Nonce` Command**) in the calculation.

All security functions are implemented using the industry-standard SHA-256 secure hash algorithm, which is part of the latest set of high-security cryptographic algorithms recommended by various government agencies and cryptographic experts. Section **SHA-256** includes a reference to the algorithm details. If desired, the SHA-256 algorithm can also be included in an HMAC sequence (See Section **`HMAC` Command**). The ATSHA204A employs full-sized, 256-bit secret keys to prevent any kind of exhaustive attack.

2. Device Organization

The device contains the following memory blocks:

- EEPROM
- SRAM

2.1 EEPROM Organization

The EEPROM contains a total of 664-bytes (5312-bits) and is divided into the following zones:

Table 2-1. ATSHA204A Zones

Zone	Description	Nomenclature
Data	Zone of 512 bytes (4.0 kb) split into 16 general purpose read-only or read/write memory slots of 32 bytes (256 bits) each that can be used to store keys, calibration data, model number, or other information, typically that relate to the item to which the ATSHA204A device is attached. Access policy of each data slot is determined by the values programmed into the corresponding configuration values. However, the policies become effective upon setting the LockValue byte only.	Slot<YY> = The entire contents stored in Slot YY of the Data zone.
Configuration	Zone of 88 bytes (704 bits) EEPROM that contains the serial number and other ID information, as well as, access the permission information for each slot of the data memory. The values programmed into the configuration zone determine the access policy of how each data slot responds. The configuration zone can be modified until it has been locked (`LockConfig` set to !=0x55). In order to enable the access policies, the LockValue byte must be set. (See section above)	SN<a:b> = A range of bytes within a field of the Configuration zone.
One Time Programmable (OTP)	Zone of 64 bytes (512 bits) of OTP bits. Prior to locking the OTPzone, the bits may be freely written using the standard `Write` command. The OTP zone can be used to store read-only data or one-way fuse type consumption logging information.	OTP<bb> = A byte within the OTP zone, while OTP<aa:bb> indicates a range of bytes.

Terms discussed within this document have the following meanings:

Table 2-2. Document Terms

Term	Meaning
Block	A single 256-bit (32-byte) area of a particular memory zone. Industry SHA-256 documentation uses the term “block” to indicate a 512-bit section of the message input. In addition, the I/O section of this document uses the term “block” to indicate a variable-length aggregate element transferred between the system and the device.
Slot	For the data zone the terms “Block” and “Slot” can be used interchangeably. For the OTP and Config zone there are multiple blocks of 32 Bytes each.
param<b>	Indicates one bit of parameter or byte field.
SRAM	Contains input and output buffers, as well as state storage locations. See Section **Static RAM (SRAM)**

On shipment from Microchip, the EEPROM contains factory test data that can be used for fixed-value board testing. This data must be overwritten with the desired contents prior to locking the configuration and/or data sections of the device. See the **Microchip website** for the document containing the specific shipment values.

2.1.1 EEPROM Data Zone

The Data zone is 512-bytes (4 kb), is part of the EEPROM array and can be used for secure storage purposes.

Prior to locking the configuration section using `Lock(Config)`, the Data zone is inaccessible and can be neither read nor written. After configuration locking, the entire Data zone can be written using the `Write` command. If desired, the data to be written can be encrypted.

In the following table, “Byte Address” is the byte address within the Data zone for the first byte in the respective slot. Because all `Reads` and `Writes` with the ATSHA204A are performed on a word (4-byte or 32-byte) basis and the word address in the table below should be used for the address parameter passed to the `Read` and `Write` commands.

Table 2-3. Data Zone Slots

Slot	Byte Address (Hex)	Word Address (Hex)	Slot	Byte Address (Hex)	Word Address (Hex)
0	0x0000	0x0000	8	0x0100	0x0040
1	0x0020	0x0008	9	0x0120	0x0048
2	0x0040	0x0010	10	0x0140	0x0050
3	0x0060	0x0018	11	0x0160	0x0058
4	0x0080	0x0020	12	0x0180	0x0060
5	0x00A0	0x0028	13	0x01A0	0x0068
6	0x00C0	0x0030	14	0x01C0	0x0070
7	0x00E0	0x0038	15	0x01E0	0x0078

2.1.2 Configuration Zone

The 88-bytes (704-bits) in the Configuration zone contain manufacturing identification data, general device and system configuration and access restriction control values for the slots within the Data zone. The values of these bytes can always be obtained using the `Read` command. The bytes of this zone are arranged as shown in the following table.

Table 2-4. Configuration Zone

Word	Byte 0	Byte 1	Byte 2	Byte 3	Default	Write Access	Read Access
0x00	SN<0:3>				01 23 xx xx	Never	Always
0x01	RevNum				xx xx xx xx	Never	Always
0x02	SN<4:7>				xx xx xx xx	Never	Always
0x03	SN<8>	Reserved	I2C_Enable	Reserved	EE 55 xx 00	Never	Always
0x04	I2C_Address	CheckMacConfig	OTP Mode	Selector Mode	C8 00 55 00	If Config Is unlocked	Always

Word	Byte 0	Byte 1	Byte 2	Byte 3	Default	Write Access	Read Access
0x05	SlotConfig 0		SlotConfig 1		8F 80 80 A1	If Config Is unlocked	Always
0x06	SlotConfig 2		SlotConfig 3		82 E0 A3 60	If Config Is unlocked	Always
0x07	SlotConfig 4		SlotConfig 5		94 40 A0 85	If Config Is unlocked	Always
0x08	SlotConfig 6		SlotConfig 7		86 40 87 07	If Config Is unlocked	Always
0x09	SlotConfig 8		SlotConfig 9		0F 00 89 F2	If Config Is unlocked	Always
0x0A	SlotConfig 10		SlotConfig 11		8A 7A 0B 8B	If Config Is unlocked	Always
0x0B	SlotConfig 12		SlotConfig 13		0C 4C DD 4D	If Config Is unlocked	Always
0x0C	SlotConfig 14		SlotConfig 15		C2 42 AF 8F	If Config Is unlocked	Always
0x0D	UseFlag 0	UpdateCount 0	UseFlag 1	UpdateCount 1	FF 00 FF 00	If Config Is unlocked	Always
0x0E	UseFlag 2	UpdateCount 2	UseFlag 3	UpdateCount 3	FF 00 FF 00	If Config Is unlocked	Always
0x0F	UseFlag 4	UpdateCount 4	UseFlag 5	UpdateCount 5	FF 00 FF 00	If Config Is unlocked	Always
0x10	UseFlag 6	UpdateCount 6	UseFlag 7	UpdateCount 7	FF 00 FF 00	If Config Is unlocked	Always
0x11	LastKeyUse 0	LastKeyUse 1	LastKeyUse 2	LastKeyUse 3	FF FF FF FF	If Config Is unlocked	Always
0x12	LastKeyUse 4	LastKeyUse 5	LastKeyUse 6	LastKeyUse 7	FF FF FF FF	If Config Is unlocked	Always
0x13	LastKeyUse 8	LastKeyUse 9	LastKeyUse 10	LastKeyUse 11	FF FF FF FF	If Config Is unlocked	Always
0x14	LastKeyUse 12	LastKeyUse 13	LastKeyUse 14	LastKeyUse 15	FF FF FF FF	If Config Is unlocked	Always
0x15	UserExtra	Selector	LockValue[1]	LockConfig	00 00 55 55	Through `UpdateExtra` Command Only	Always

Note:

1. LockValue was previously known as LockData.

2.1.2.1 I2C_Enable

Bit 7–1: Ignored and set by Microchip.

Bit 0: 0 = Single-Wire Interface Mode.
1= I²C interface Mode.

2.1.2.2 I2C_Address

I²C Mode I2C_Enable<0> = 1

Bits 7 – 1: I²C device address

Bit 3: TTL Enable (Dual purpose bit)
Part of I²C Address and set's the threshold level.
0= Input level uses a fixed reference.
1 = Input level uses the V_{CC} as reference.

Bit 0: Ignored.

Single-Wire Mode I2C_Enable<0> = 0

Bits 7–4: Ignored.

Bit 3: TTL Enable
0= Input level uses a fixed reference.
1 = Input level uses the V_{CC} as reference.

Bits 2–0: Ignored.

2.1.2.3 CheckMacConfig

This byte applies only to the CheckMac, Read and Write commands:

- **Read and Write:** CheckMacConfig<0> controls Slots 0 and 1, CheckMacConfig<1> controls Slots 2 and 3 and so on. Any encrypted Read or Write command fails if the value in TempKey.SourceFlag does not match the corresponding bit in this byte. This byte is ignored for clear text reads and writes.
- **CheckMac:** CheckMacConfig<0> controls slot 1, CheckMacConfig<1> controls Slot 3 and so on. The copy function can only be enabled if the CheckMacSource value corresponding to the target slot matches the value of Mode bit 2 of the CheckMac command. The command fails if Mode bit 2 does not match TempKey.SourceFlag, so this is equivalent to requiring the corresponding bit in this byte to match TempKey.SourceFlag.

2.1.2.4 OTP Mode

0xAA (Read-only mode) = When OTP zone is locked, writes are disabled and reads of all words are permitted.

0x55 (Consumption mode) = Writes to the OTP zone when the OTP zone is locked causes the bits to transition only from a one to a zero. Reads of all words are permitted.

0x00 (Legacy mode) = When OTP zone is locked, writes are disabled, reads of Words 0 and 1 and 32-byte reads are disabled.

All other modes are reserved.

2.1.2.5 Selector Mode

If `0x00`, then the Selector is updated with `UpdateExtra`.

All other values can only allow the Selector to be updated if its value is zero.

2.1.2.6 Slot Config

See Table **SlotConfig Bits (Per Slot)**.

2.1.2.7 UseFlag

For uses with “limited-use slots”. The quantity of “`1`” bits represents the number of times that slots 0 thru 7 may be used before being disabled.

2.1.2.8 UpdateCount

Indicates how many times slots 0 through 7 have been updated with `DeriveKey`.

2.1.2.9 LastKeyUse

Used to control limited use for Slot 15. Each “`1`” bit represents a remaining use for Slot 15. Applies only if SlotConfig<5> LimitedUse is set.

2.1.2.10 UserExtra

For general system use, can be modified through the `UpdateExtra` command.

2.1.2.11 Selector

Selects which device remains in active mode after the execution of the `Pause` command.

2.1.2.12 LockValue

Controls the Data and OTP zones are unlocked and can be freely written but not read.

`0x55` = The Data and OTP zones are unlocked and has write access.

`0x00` = The Data and OTP zones are locked and take on the access policies defined in the configuration zone. Slots in the Data zone can only be modified based on the corresponding WriteConfig fields. The OTP zone can only be modified based on the OTP mode.

2.1.2.13 LockConfig

Configuration zone access.

`0x55` = The Configuration zone has write access (unlocked).

`0x00` = The Configuration zone does not have write access (locked).

2.1.2.14 SlotConfig (Bytes 20 – 51)

The 16 SlotConfig elements configure the access protections for each of the 16 slots within the ATSHA204A. Each configuration element consists of 16 bits, which control the usage and access for that particular slot or key. The SlotConfig field is interpreted according to the table below when the Data zone is locked. When the Data zone is unlocked, these restrictions do not apply and all slots may be freely written and none may be read.

Table 2-5. SlotConfig Bits (Per Slot)

Bit	Name	Description
15-12	WriteConfig	See detailed function definition for use.
11-8	WriteKey	Slot of the key to be used to validate encrypted writes.

Bit	Name	Description
7	IsSecret	**0 =** The slot is not secret and allows clear read, clear write, no MAC check and no `Derivekey` Command. **1 =** The slot is secret. Reads and writes if allowed, must be encrypted.
6	EncryptRead	**0 =** Clear reads are permitted. **1 =** Requires the slot to be Secret and encrypted read to access.
5	LimitedUse(1)	**0 =** No limit on the number of time the key can be used. **1 =** Limit on the number of time the key can be used based on the UseFlag (or LastKeyUse) for the slot.
4	CheckOnly	**0 =** This slot can be used for all crypto commands. **1 =** This slot can only be used for `CheckMac` and `GenDig` followed by `CheckMac` Commands.
3-0	ReadKey	Slot of the key to be used for encrypted reads. If `0x0`, then this slot can be used as the source slot for the `CheckMac/Copy` Command.

Note:

1. LimitedUse bit was previously named SingleUse.

Table 2-6. Write Configuration Bits — `Derivekey` Command

Bit 15	Bit 14	Bit 13	Bit 12	Source Key(1)	Description
0	X	1	0	Target	`DeriveKey` command can be run without authorizing MAC (Roll).
1	X	1	0	Target	Authorizing MAC required for `DeriveKey` command (Roll).
0	X	1	1	Parent	`DeriveKey` command can be run without authorizing MAC (Create).
1	X	1	1	Parent	Authorizing MAC required for `DeriveKey` command (Create).
X	X	0	X	—	Slots with this value in the WriteConfig field may not be used as the target of the `DeriveKey` command.

Note:

1. The source key for the computation performed by the `DeriveKey` command can either be the key directly specified in Param2 (the “Target”) or the key at SlotConfig<Param2>. WriteKey (the “Parent”).
   See Section **Key Values** for more details.

Table 2-7. Write Configuration Bits — `Write` Command

Bit 15	Bit 14	Bit 13	Mode Name	Description
0	0	0	Always	Clear text writes are always permitted on this slot. Slots set to “always” should never be used as key storage. Either 4 or 32 bytes may be written to this slot.
X	0	1	Never	Writes are never permitted on this slot using the `Write` command Slots set to “never” can still be used as key storage.
1	0	X	Never	Writes are never permitted on this slot using the `Write` command Slots set to “never” can still be used as key storage.
X	1	X	Encrypt	Writes to this slot require a properly computed MAC and the input data must be encrypted by the system with WriteKey using the encryption algorithm documented in the `Write` command description Section (8.5.18 Write Command). 4-byte writes to this slot are prohibited.

The 4-bit WriteConfig field is interpreted by the `Write` command as shown in Table **Write Configuration Bits —Write Command**, where X means don’t care.

The tables overlap. For example, a code of `0b0110` indicates that a slot can be written in encrypted form by using the `Write` command and it can also be the target of an unauthorized `DeriveKey` command with the target as the source.

The IsSecret bit controls internal circuitry necessary for proper security for slots in which reads and/or writes must be encrypted or are prohibited altogether. It must also be set for all slots that are to be used as keys, including those created or modified with `DeriveKey`. Specifically, to enable proper device operation, this bit must be set unless WriteConfig is “Always”. 4-byte accesses are prohibited to/from slots in which this bit is set.

Slots used to store key values should always have IsSecret set to one and EncryptRead set to zero (reads prohibited) for maximum security. For fixed key values, WriteConfig should be set to “Never”. When configured in this way, there is no way to read or write the key after the Data zone is locked. It may only be used for crypto operations.

Some security policies require secrets to be updated from time to time. The ATSHA204A supports this capability in the following way:

- WriteConfig for the particular slot should be set to “Encrypt” and SlotConfig.WriteKey should point back to the same slot by setting WriteKey to the slot ID. A standard `Write` command can be then used to write a new value to this slot provided that the authentication MAC is computed using the old (current) key value.

2.1.2.15 Special Memory Values in the Configuration Zone (Bytes 0 – 12)

Various fixed information is included in the ATSHA204A that can never be written under any circumstances and can always be read, regardless of the state of the lock bits.

- **SerialNum**
  Nine bytes (SN<0:8>) which together form a unique value that is never repeated for any device in the CryptoAuthentication family. The serial number is divided into two groups:
  1.1. **SN<0:1> and SN<8>**
  The values of these bits are fixed at manufacturing time in most versions of the ATSHA204A. Their default value is (`0x01 0x23 0xEE`). These 24 bits are always included in the SHA-256 computations made by the ATSHA204A.

1.2. **SN<2:7>**
The values of these bits are programmed by Microchip during the manufacturing process and are different for every die. These 6-bytes (48-bits) are optionally included in some SHA-256 computations made by the ATSHA204A

- **RevNum**
Four bytes of information that are used by Microchip to provide manufacturing revision information. These bytes can be freely read as RevNum<0:3>, but should never be used by system software, because they may change due to a silicon revision.

2.1.3 One Time Programmable (OTP) Zone

The OTP zone of 64 bytes (512 bits) is part of the EEPROM array and can be used for read-only storage.

Prior to locking the configuration section using `Lock(LockConfig)`, the OTP zone is inaccessible and can be neither read nor written. After configuration locking, but prior to locking of the OTP zone using `Lock(LockValue)`, the entire OTP zone can be written using the `Write` command. If desired, the data to be written can be encrypted. When unlocked the OTP zone cannot be read.

Once the OTP zone is locked, the OTP mode byte in the Configuration zone controls the permissions of this zone, as follows:

- **Read-only Mode**
The data cannot be modified and would be used to store fixed model numbers, calibration information, manufacturing history and/or other data that should never change. The `Write` command always returns an error and leaves the memory unmodified. All 64-bytes within the OTP section are always available for reading using either 4-byte or 32-byte reads.
- **Consumption Mode**
The bits function as one-way fuses and can be used to track consumption or usage of the item to which the ATSHA204A is attached. For examples, in a battery, they might be used to track charging cycles or use time; in a printer ink cartridge, they might track the quantity of material consumed; in a medical device, they might track the number of permitted uses for a limited use item. In this mode, the `Write` command can only cause bits to transition from a one to a zero. Logically, this means the data value in the input parameter list is AND'ed with the current value in the word(s) and the result written back to memory. As an example, writing a value of `0xFF` results in no change to the byte and writing a value of `0x00` causes the byte in memory to go to zero, regardless of the previous value. Once a bit has transitioned to a zero, it can never transition back to a one.
- **Legacy Mode**
The operation of the OTP zone is consistent with the fuse array on the Microchip(Formerly Atmel) ATSA102S. Reads of words zero and one are always prohibited, while reads of the remaining 14 words are always permitted. Only 4-byte (32-bit) reads are permitted and any attempt to execute a 32-byte (256-bit) read results in an error return code. All Write operations to the OTP zone are prohibited. See Section 9. Compatibility for more of the Microchip ATSA102S compatibility details.

All OTP zone bits have a value of one on shipment from the Microchip factory.

Table 2-8. OTP Zone

Word (HEX)	Address (HEX)	Default
0x00	0x00	0xFFFFFFFF
0x01	0x04	0xFFFFFFFF
0x02	0x08	0xFFFFFFFF

Word (HEX)	Address (HEX)	Default
0x03	0x0C	0xFFFFFFFF
0x04	0x10	0xFFFFFFFF
0x05	0x14	0xFFFFFFFF
0x06	0x18	0xFFFFFFFF
0x07	0x1C	0xFFFFFFFF
0x08	0x20	0xFFFFFFFF
0x09	0x24	0xFFFFFFFF
0x0A	0x28	0xFFFFFFFF
0x0B	0x2C	0xFFFFFFFF
0x0C	0x30	0xFFFFFFFF
0x0D	0x34	0xFFFFFFFF
0x0E	0x38	0xFFFFFFFF
0x0F	0x3C	0xFFFFFFFF

2.1.4 Device Locking

There are two separate lock bytes for the device:

- One to lock the configuration zone (that is controlled by LockConfig, byte 87).
- One to lock both the Data and OTP zones (that are controlled by LockValue, byte 86). This enables the access polices for each Data zone slot based on the Slot configuration.

These locks are stored within separate bytes in the Configuration zone and can be modified only through the `Lock` command. After a memory zone is locked, there is no way to unlock it. Locking of the Data/OTP zone does not mean the slots can not be modified. The slots can be modified based on the access policies defined by the Slot configuration.

The device should be personalized at the system manufacturer with the desired configuration information and the Configuration zone should be locked. When this lock is complete, all necessary writes of public and secret information into the EEPROM slots should be performed using encrypted writes if appropriate. Upon completion of writes to the data and OTP zones, the Data and OTP zones the LockValue byte should be written.

It is vital that the LockValue byte be set to lock prior to release of the system containing the device into the field in order to protect the data stored in the Data and OTP zones. Failure to lock these zones may permit modification of any secret keys and may lead to other security problems.

Any attempt to read or write the Data or OTP sections prior to locking the configuration section causes the device to return an error.

Contact Microchip for optional secure personalization services.

2.1.4.1 Configuration Zone Locking

Certain bytes within the configuration zone cannot be modified, regardless of the state of LockConfig. Access to the remainder of the bytes within the zone is controlled using the LockConfig byte in the configuration zone, as shown in the table below. Throughout this document, if LockConfig is `0x55`, then the configuration zone is said to be unlocked; otherwise it is locked.

Table 2-9. Configuration Zone Locking

Lock State	Read Access	Write Access
LockConfig == 0x55 (unlocked)	Read	Write
LockConfig != 0x55 (locked)	Read	<never>

2.1.4.2 Data and OTP Zone Locking

Throughout this document, if LockValue is 0x55, then both the Data and OTP zones are said to be unlocked; otherwise they are locked.

There is neither read nor write access to the Data and OTP zones prior to locking of the Configuration zone.

Table 2-10. Data and OTP Zone Access Restrictions

Lock State	Read Access	Write Access
LockValue == 0x55 (unlocked)	<never>	Write
LockValue != 0x55 (locked)	Read(1)	Write(1)

Note:

1. Based on Slot Configuration for a given slot.

2.1.4.3 OTP Zone Locking

Reads and writes of the OTP zone depend upon the state of the LockConfig, LockValue and OTP mode bytes in the Configuration zone.

2.2 Static RAM (SRAM)

The device includes an SRAM array that is used to store the input command or output result, intermediate computation values and/or an ephemeral key. The entire contents of this memory are always invalidated whenever the device goes into sleep mode or the power is removed. The ephemeral key is named TempKey and can be used as an input to the MAC, HMAC, CheckMac, GenDig and DeriveKey commands. It is also used as the Data protection (Encryption or Decryption) key by the Read and Write commands. See Section **TempKey**.

2.2.1 TempKey

TempKey is a storage register in the SRAM array that can be used to store an ephemeral result value from the Nonce, GenDig, CheckMac, or SHA commands. The contents of this register can never be read from the device (although the device itself can read and use the contents internally).

This register contains the elements shown in the table below.

Table 2-11. TempKey Storage Register

Name	Bit Length	Description
TempKey	256 (32-bytes)	Nonce (from Nonce command) or Digest (from GenDig command).
SlotID	4	If TempKey was generated by GenDig (see the GenData and CheckFlag bits), these bits indicate which key was used in its computation. The four bits represent one of the slots of the Data zone.

Name	Bit Length	Description
SourceFlag	1	The source of the randomness in TempKey: **0 =** Internally generated random number (Rand). **1 =** Input seed only, no internal random generation (Input).
GenData	1	**0 =** TempKey.SlotID is not meaningful and is ignored. **1 =** The contents of TempKey were generated by `GenDig` using one of the slots in the Data zone (and TempKey.SlotID is meaningful).
CheckFlag	1	**0 =** TempKey contents have been generated using a Nonce, SHA or GenDig without a CheckMac key restriction. **1 =** The contents of TempKey were generated by the `GenDig` command and at least one of the keys used in that generation is restricted to the `CheckMac` command (SlotConfig.CheckOnly is one)
Valid	1	**0 =** The information in TempKey is invalid. **1 =** The information in TempKey is valid.

In this specification, the name “TempKey” refers to the contents of the 32-byte (256-bit) Data register. The remaining bit fields are referred to as TempKey.SourceFlag, TempKey.GenData and so on.

The TempKey.Valid bit is cleared to zero under any of the following circumstances:

- Power-up, sleep, brown-out, watchdog expiration, or tamper detection. The contents of TempKey are however retained when the device enters idle mode.
- After the execution of any command other than `Nonce` or `GenDig`, regardless of whether or not the command execution succeeds. It may be cleared by the `CheckMac` command unless a successful copy takes place. It is not cleared if there is a communications problem, as evidenced by a Cyclic Redundancy Check (CRC) error.
- An error during the parsing or execution of a `GenDig` and/or `Nonce` command.
- Execution of `GenDig` replaces any previous output of the `Nonce` command with the output of the `GenDig` command. Execution of the `Nonce` command likewise replaces any previous output of the `GenDig` command.

3. Security Features

3.1 Physical Security

The ATSHA204A incorporates a number of physical security features designed to protect the EEPROM contents from unauthorized exposure. The security measures include:

- An Active Shield Over the Part
- Internal Memory Encryption
- Secure Test Modes
- Glitch Protection
- Voltage Tamper Detection
- Temperature Tamper Detection

Pre-programmed transport keys stored on the ATSHA204A are encrypted in such a way as to make retrieval of their values using outside analysis very difficult.

Both the logic clock and logic supply voltage are internally generated, preventing any direct attack on these two signals using the pins of the device.

3.2 Random Number Generator (RNG)

The ATSHA204A includes a high-quality RNG that returns a 32-byte random number to the system. The device combines this generated number with a separate input number to form a nonce that is stored within the device in TempKey and may be used by subsequent commands.

The system may use this RNG for any purpose. One common purpose would be as the input challenge to the `MAC` command on a separate CryptoAuthentication device. The device provides a special random command for such purposes, which does do not affect the internally stored nonce.

To simplify system testing, prior to locking the Configuration zone the RNG always returns the following 32 byte value:

`0xFF FF 00 00 FF FF 00 00 …`

where `0xFF` is the first byte read from the device and is used for the SHA message.

To prevent replay attacks on encrypted data that is passed to or from the ATSHA204A, the device requires that a new, internally generated nonce be included as part of the encryption sequence used to protect the data being read or written. To implement this requirement, the data protection key generated by `GenDig` and used by the `Read` or `Write` command must use the internal RNG during the creation of the nonce.

Random numbers are generated from a combination of the output of a hardware RNG and an internal seed value, which is not externally accessible. The internal seed is stored in the EEPROM and is normally updated once after every power-up or sleep/wake cycle. After the update, this seed value is retained in SRAM registers within the device that are invalidated if the device enters sleep mode or the power is removed.

4. General I/O Information

Communication with the ATSHA204A is achieved through one of two different protocols (I^2C or Single-Wire) and is selected based on the device ordered:

- **Single-Wire Interface**
  Uses a single GPIO connection on the system microprocessor connected to the SDA pin on the device. It permits the fewest number of connector pins to any removable/replaceable entity. The bit rate is up to 25.6 kb/s and is compatible with standard UART signaling.
- **I^2C Interface**
  This mode is compatible with the Microchip AT24C16 Serial EEPROM interface. Two pins are required, Serial Data (SDA) and Serial Clock (SCL). The I^2C interface supports a bit rate of up to 1 Mb/s.

The lowest levels of the I/O protocols are described in Section **Single-Wire Interface** and Section **I^2C Interface**. On top of the I/O protocol level, both interfaces transmit exactly the same bytes to and from the device to implement the cryptographic commands and error codes documented in Section **Security Commands**.

The device implements a failsafe internal watchdog timer that forces it into a very low-power mode after a certain time interval, regardless of any current activity. System programming must take this into consideration. See Section **Watchdog Failsafe** for details.

4.1 Byte and Bit Ordering

CryptoAuthentication devices use a common ordering scheme for bytes and also for the way in which numbers and arrays are represented in this datasheet:

- All multi-byte aggregate elements are treated as arrays of bytes and are processed in the order received or transmitted with index #0 first.
- 2-byte (16-bit) integers, typically Param2 appear on the bus LSB first.

The bit order is different depending on the I/O channel used:

- On the Single-Wire Interface, data is transferred to/from the ATSHA204A LSb first on the bus.
- On the I^2C Interface, data is transferred to/from the ATSHA204A MSb first on the bus.

4.1.1 Output Example

The following bytes are returned in this order on the bus by a 32-byte read of the configuration section with an input address of `0x0000`:

SN<0>, SN<1>, SN<2>, SN<3>, RevNum<0>, RevNum<1>, RevNum<2>, RevNum<3>, SN<4>, SN<5>, SN<6>, SN<7>, SN<8>, reserved, I2C_Enable, reserved, I2C_Address, OTPmode, SelectorMode, SlotConfig<0>.Read, SlotConfig<0>.Write, SlotConfig<1>.Read, SlotConfig<1>.Write, SlotConfig<2>.Read, SlotConfig<2>.Write, SlotConfig<3>.Read, SlotConfig<3>.Write, SlotConfig<4>.Read, SlotConfig<4>.Write, SlotConfig<5>.Read, SlotConfig<5>.Write

4.1.2 MAC Message Example

The following bytes are passed to the SHA engine for a `MAC` command using a mode value of 0x71 and a SlotID of slot x. In the example below, K<x> indicates the SlotID of slot x in the Data zone, with K<0> being the first byte on the bus for a read from or write to that slot. OTP<0> indicates the first byte on the bus for a read of the OTP zone at address zero and so on.

K<0>, K<1>, K<2>, K<3> … K<31>, TempKey<0>, TempKey<1>, TempKey<2>, TempKey<3> … TempKey<31>, Opcode (=0x08), Mode (=0x71), Param2(LSB = 0xYY), Param2(MSB = 0x00), OTP<0>, OTP<1>, OTP<2>, OTP<3>, OTP<4>, OTP<5>, OTP<6>, OTP<7>, OTP<8>, OTP<9>, OTP<10>, SN<8>, SN<4>, SN<5>, SN<6>, SN<7>, SN<0>, SN<1>, SN<2>, SN<3>.

For more details regarding MAC messages, see Section **MAC Command**.

5. Single-Wire Interface

In the Single-Wire Interface mode, communications to and from the ATSHA204A take place over the SDA pin, a single, asynchronously timed wire and the SCL pin is ignored.

The sleep current specification values are guaranteed only if the SCL pin is held low or left unconnected.

The overall communications structure is a hierarchy: The table below shows the tokens used for the Single-Wire Interface with a standard RS-232 port. The Host UART port should be set to 7-bit data words and 230.4 kBaud data rate.

Table 5-1. Wake and I/O Tokens

Token Type	Token Value	Start (1)	Wake Token LSb: MSb							Stop (1)
			b0	b1	b2	b3	b4	b5	b6	
Wake (2)	0x00	0	0	0	0	0	0	0	0	1
Logic 0 (3)	0x7D	0	1	0	1	1	1	1	1	1
Logic 1 (3)	0X7F	0	1	1	1	1	1	1	1	1

Note:

1. All Tokens must be preceded by a LOW Start Pulse to synchronize the data capture and end with a HIGH Stop value.
2. A Wake Token creates a low pulse great enough to wake up the device.
3. Logic 0, Logic 1 I/O tokens represent a single bit of data. 8 I/O tokens would be needed to create a single byte of data.

I/O Flags - Flags consist of eight tokens (bits) that convey the direction and meaning of the next group of bits (if any) that may be transmitted. Flags are always transmitted LSb first.

Blocks - Blocks of data follow the command and transmit flags. They incorporate both a byte count and a checksum to ensure proper data transmission.

Packets - Packets of bytes form the core of the block (minus the byte count and CRC). They are either the input or output parameters of a CryptoAuthentication command or status information from the ATSHA204A.

5.1 I/O Tokens

There are a number of I/O tokens that may be transmitted over the Single-Wire Interface:

- **Input** (to the ATSHA204A)
  - Wake: wake the device up from either sleep or idle states.
  - Zero: send a single bit from the system to the device with a value of zero.
  - One: send a single bit from the system to the device with a value of one.
- **Output** (from the ATSHA204A)
  - ZeroOut: send a single bit from the device to the system with a value of zero.
  - OneOut: send a single bit from the device to the system with a value of one.

The waveforms are the same in either direction. There are some differences in timing; however, based on the expectation that the Host has a very accurate and consistent clock, while the ATSHA204A has

significant part-to-part variability in its internal clock generator, due to normal manufacturing and environmental fluctuations.

The bit timing is designed to permit a standard UART running at 230.4 kBaud to transmit and receive the tokens efficiently. Each byte transmitted or received by the UART corresponds to a single bit received or transmitted by the device. The UART needs to be configured with 7-bits of data having `0x7F` corresponding to a Logic `1` and `0x7D` corresponding to a Logic `0`.

The Wake token is special in that it requires an extra long low pulse of t_{WLO} on the SDA pin (see Table **AC Parameters – All I/O Interfaces**), which cannot be confused with the shorter low pulses that occur during a Data token (Zero, One, ZeroOut, or OneOut). Devices that are in either the idle or sleep state ignore all data tokens until they receive a legal Wake token. Do not send a Wake token to devices that are awake, as they lose synchronization because the waveform can be resolved to neither a legal one nor zero. See Section **Synchronization Procedures** for the procedure to regain synchronization.

5.2 I/O Flags

The system is always the bus master; so before any I/O transaction, the system must send an 8-bit flag to the device to indicate the I/O operation to be subsequently performed, as shown in the table below.

Table 5-2. I/O Flags

Name	Value	Meaning
Sleep (low-power)	`0xCC`	The ATSHA204A goes into the low-power sleep mode and ignores all subsequent I/O transitions until the next Wake flag. The entire volatile state of the device is reset.
Idle	`0xBB`	The ATSHA204A goes into the idle state and ignores all subsequent I/O transitions until the next Wake flag. The contents of TempKey and RNG seed registers are retained.
Command	`0x77`	Write subsequent bytes to sequential addresses in the input command buffer.
Reserved	All Other Values	These flags should not be sent to the device.
Transmit	`0x88`	Communicates to the device to wait for a bus turnaround time and then start transmitting its response to the previously transmitted command block. When valid data is in the output buffer, the transmit flag may be repeatedly issued to the device to resend the buffer to the system.
Wake	See Interface	Wake the device from low-power mode and reset the watchdog counter.

5.2.1 Transmit Flag

The transmit flag is used to turn the bus around so that the ATSHA204A can send data back to the system. The bytes that the device returns to the system depend upon the current state of the device and may include either status, error code, or command results.

When the device is busy executing a command, it ignores the SDA pin and any flags that are sent by the system. See Section **Command Opcodes, Short Descriptions and Execution Times** for execution delays in the device for each command type. The system must observe these delays before trying to communicate with the device after sending a command.

5.3 Synchronization

Because the communications protocol is half-duplex, there is the possibility that the system and the ATSHA204A can fall out of synchronization with each other. In order to speed recovery, the device implements a timeout that forces it to sleep under certain circumstances.

5.3.1 I/O Timeout

After a leading transition for any data token has been received, the ATSHA204A expects the remaining bits of the token to be properly received by the device within the $t_{TIMEOUT}$ interval. Failure to send enough bits or the transmission of an illegal token (a low pulse exceeding t_{ZLO}) causes the device to enter the sleep state after the $t_{TIMEOUT}$ interval.

The same timeout applies during the transmission of the command block. After the transmission of a legal command flag, the I/O timeout circuitry is enabled until the last expected data bit is received.

Note: The Timeout Counter is reset after every legal token and the total time to transmit the command may exceed the $t_{TIMEOUT}$ interval while the time between bits may not.

The I/O timeout circuitry is disabled when the device is busy executing a command.

5.3.2 Synchronization Procedures

If the device is not busy when the system sends a transmit flag, the device should respond within $t_{TURNAROUND}$. If t_{EXEC} time has not already passed, the device may be busy and the system should poll or wait until the maximum t_{EXEC} time has elapsed. If the device still does not respond to a second transmit flag within $t_{TURNAROUND}$, it may be out of synchronization. At this point, the system may take the following steps to reestablish communication:

1. Wait $t_{TIMEOUT.}$
2. Send the transmit flag.
3. If the device responds within $t_{TURNAROUND}$, then the system may proceed with more commands.
4. Send a Wake token.
5. Wait $t_{WHI.}$
6. Send the transmit flag.
7. The device should respond with a `0x11` status within $t_{TURNAROUND}$, at which time system may proceed with commands.

Any command results in the I/O buffer may be lost when the system and device lose synchronization.

5.4 Sharing the Interface

Multiple CryptoAuthentication devices may share the same interface, as follows:

1. System issues a Wake token (Section **Watchdog Failsafe**) to wake-up all devices.
2. The system issues the `Pause` command to put all but one of the devices into idle mode. Only the remaining device then sees any commands that the system sends. When the system has completed talking to the one active device, it sends an idle flag, which the idle devices ignore, but puts the single remaining active device into the idle mode. See Section **`Pause` Command** for more details.

Steps 1 and 2 are repeated for each device on the wire. If the system has completed communications with the final device, it should wake all the devices up and then put all the devices to sleep to reduce total power consumption.

The device uses the selector byte within the configuration zone to determine which device stays awake. Only that device with a selector value that matches the input parameter of the `Pause` command stays awake. In order to facilitate late configuration of systems that use the multi-device sharing mode, the following three update capabilities for the selector byte are supported:

1. **Unlimited Updates**
   At any time, the `UpdateExtra` command can be executed to write the value in the selector field of the Configuration zone. To enable this mode, set the SelectorMode byte in the Configuration zone to zero.
2. **One-time Field Update**
   If the SelectorMode byte is set to a non-zero value and the selector byte is set to a zero value prior to locking the Configuration zone. Then, at any time after the Configuration zone is locked the `UpdateExtra` command can be used one time to set Selector to a non-zero value. The `UpdateExtra` command is not affected by the LockValue byte.
3. **Fixed Selector Value**
   The selector byte can never be modified after the Configuration zone is locked if both SelectorMode and Selector are set to non-zero values. The `UpdateExtra` command always returns an error code.

5.5 Transaction Example

Wake (Single-Wire)		
Host		Device
Wake	→	
Transmit	→	
	←	Data

Example (Single-Wire)		
Host		Device
Wake	→	
Transmit	→	
	←	Data
Command	→	
Data	→	
Transmit	→	
	←	Data
Idle/Sleep	→	

Table 5-3. Example (Single-Wire)

	Wake Token 0x00								Transmit 0x88								Count 0x04								Status 0x11							
Host									0	0	0	1	0	0	0	1																
Device																	0	0	1	0	0	0	0	0	1	0	0	0	1	0	0	0