Application Note

Command Sequences

Atmel ATSHA204

Introduction

This application note provides example command sequences for the Atmel® ATSHA204 device. These sequences attempt to cover the normal use cases for the ATSHA204 in terms of the expected command sequences that would be necessary to accomplish the task.

Overview

The ATSHA204 device is a highly configurable security IC. There are a number of ways to configure the device based on the use case that is in your system.

In this application note, the Client is considered the entity which needs to be authenticated, and the Host is the system which is controlling the process attempting to authenticate the Client device. In most examples, it is expected that the Host contains an ATSHA204 device; however, in some cases, the Host may just be a software sequence as specified.

See application notes, “Atmel ATSHA204 Personalization Guide” and “How to Personalize the Atmel ATSHA204” for more information.

1. Fixed Challenge Authentication

1.1 MAC

Step	To Client ATSHA204	To Host ATSHA204
1	Nonce (Fixed)	
2	MAC (eeKey, TempKey)	
3		CheckMac (eeKey, Input 1, Output 2)

Step	To Client ATSHA204	To Host ATSHA204
1	Nonce (Fixed)	
2	MAC (eeKey, TempKey)	
3		Nonce (Fixed — Step 1)
4		CheckMac (eeKey, TempKey, Output 2)

1.2 HMAC

Step	To Client ATSHA204	To Host ATSHA204
1	Nonce (Fixed)	
2	HMAC (eeKey, TempKey)	
3		Compute Nonce as in Step 1.
4		Compute HMAC as in Step 2. Compare results.

1.3 Single Use

Same as MAC and HMAC, but the key is Single Use in Slot 0 to Slot 7, slotConfig.singleUse is one. The test can be run nine times (the ninth test should fail), or UseFlag can be initialized to 0x01; the test can be run two times (the second test should fail).

1.4 Limited Use

Same as MAC and HMAC, but the key is Limited Use and must be in Slot 15, slotConfig.singleUse is one. The test can be run 129 times (the 129th test should fail), or LastKeyUse can be initialized to 0x00 00… 01; the test can be run two times (the second test should fail).

1.5 Challenge in the Input Stream

Step	To Client ATSHA204	To Host ATSHA204
1	MAC (eeKey, Fixed)	
2		CheckMac (eeKey, Input 1, Output 2)

1.6 Challenge in the Input Stream — Key in TempKey

Similar to challenge in the input stream, but using TempKey generated by GenDig instead of eeKey.

Step	To Client ATSHA204	To Host ATSHA204
1	Nonce (Fixed)	
2	GenDig (eeKey)	
3	MAC (TempKey, Challenge)	
4		Nonce (Fixed)
5		GenDig (eeKey)
6		CheckMac (TempKey, Challenge, Output 3)

2. Random Challenge Authentication

2.1 Fixed Keys — MAC

Step	To Client ATSHA204	To Host ATSHA204
1		Random()
2	Nonce (Random, Seed from Step 1)	
3	MAC (eeKey, TempKey)	
4		CheckMac (eeKey, SHA (I/O Step 2), Output 2)

2.2 Fixed Keys — Diversified Response and MAC

Step	To Client ATSHA204	To Host ATSHA204
1		Random()
2	Nonce (Random, Seed from Step 1)	
3	MAC (UseSN, eeKey, TempKey)	
4		CheckMac (eeKey, SHA(I/O Step 2), Output 2, SN)

2.3 Fixed Keys — HMAC

Step	To Client ATSHA204	System Software
1	Nonce (Random)	
2	HMAC (eeKey, TempKey)	
3		Calculate Nonce using output of Step 1.
4		Compute HMAC as in Step 2. Compare results.

2.4 Diversified Keys — Ephemeral (TempKey) on Host

Step	To Client ATSHA204	To Host ATSHA204
1		Random()
2	Nonce (Random, Seed from Step 1)	
3	MAC (eeKey and TempKey)	
4		Nonce (Fixed, Client SN)
5		GenDig (RootKey)
6		CheckMac (TempKey, SHA(I/O Step 2), Output 3)

2.5 Diversified Keys, Fixed Nonce at Client

Step	To Client ATSHA204	To Host ATSHA204
1		Random()
2	Nonce (Fixed, Rand from Step 1)	
3	MAC (eeKey, TempKey)	
4		Nonce (Fixed, Client SN)
5		GenDig (RootKey)
6		CheckMac (TempKey, Random f/Step 1, Output 3)

2.6 DiversifieD Keys, New Special Nonce Mode

Step	To Client ATSHA204	To Host ATSHA204
1		Random()
2	Nonce (Random, Output 1)	
3	MAC (eeKey, TempKey)	
4		Nonce (Fixed, Random from step 2)
5		Nonce (Calculate, Random from step 1)
6		Nonce (Fixed, Client SN)
7		GenDig (RootKey)
8		CheckMac (TempKey, Output 5, Output 3)

2.7 Diversified Keys, Static (Stored in EE Slot) on Host

Step	To Client ATSHA204	To Host ATSHA204
1		Random()
2	Nonce (Random, Seed from Step 1)	
3	MAC (eeKey, TempKey)	
4		Nonce (Fixed, Client SN)
5		DeriveKey (eeKey)
6		CheckMac (eeKey, SHA(I/O Step 2), Output 3)

3. Child Key

3.1 Normal

The Child Key can be any key but slotConfig.writeKey. For the Child Key should point to the Parent Key. slotConfig.writeConfig for the Child Key should be 0x11. Requires CheckOnly to be set for the hostParent, which must have the same value as the Client parent but *must* have the CheckOnly bit set to permit OtherData to pass to GenDig.

Step	To Client ATSHA204	To Host ATSHA204
1	Nonce (Fixed)	
2	DeriveKey (Fixed, Child)	
3	Nonce (Fixed)	
4	MAC (eeChild, TempKey)	
5		Nonce (Fixed)
6		GenDig (hostParent)
7		CheckMac (Input 3, TempKey, Output 4)

3.2 Authenticated

Same as Normal, slotConfig.writeConfig for the Child Key should be 1x11, and the necessary authentication should be calculated externally.

3.3 Single Use Parent

Same as Normal, but the Parent Key is Single Use. The Parent Key is in Slot 0 to Slot 7, and slotConfig.singleUse is one. The test can be run nine times (the ninth test should fail), or UseFlag can be initialized to 0x01; the test can be run two times (the second test should fail). The Child Key is *not* use limited — can be any key.

3.4 Single Use Child

Same as Normal, but the Child Key is Single Use and *must* be in Slot 0 to Slot 7, slotConfig.singleUse is one. MAC can be repeated nine times (the ninth MAC should fail).

Caution: UseFlag *cannot* be initialized since DeriveKey will overwrite it.

3.5 Limited Use

Same as Normal, but the parent key is Limited Use and must be in Slot 15, slotConfig.singleUse is one. The test can be run 129 times (the 129th test should fail), or LastKeyUse can be initialized to 0x00 00… 01; the test can be run two times (the second test should fail). Child Key is *not* use limited — can be any key.

4. Roll Key

4.1 Normal

The Target Key can be any key. slotConfig.writeConfig for key should be 0x10. Requires CheckOnly to be set for the hostKey, which must have the same value as the Client Key but *must* have the CheckOnly bit set to permit OtherData to passed to GenDig.

Step	To Client ATSHA204	To Host ATSHA204
1	Nonce (Fixed)	
2	DeriveKey (Fixed, eeTarget)	
3	Nonce (Fixed)	
4	MAC (eeTarget, TempKey)	
5		Nonce (Fixed)
6		GenDig (hostKey)
7		CheckMac (Input 3, TempKey, and Output 4)

4.2 Authenticated

Same as Normal, but slotConfig.writeConfig for the Child Key should be 1x10, and the necessary authentication should be calculated externally.

4.3 Single Use

Same as Normal, but the key is Single Use. The key must be in Slot 0 to Slot 7, slotConfig.singleUse is one. In this case, the Nonce/MAC pair should be run nine times (the ninth Nonce/MAC should fail).

Caution: UseFlag cannot be initialized since DeriveKey will overwrite it.

4.4 Separate Key Authentication (CheckMac), No MAC

Step	To Client ATSHA204	To Host ATSHA204
1	Nonce (Fixed)	
2	CheckMac (authKey, Nonce from Step 1, SHA (authKey Output 1)	
3	Nonce (Fixed)	
4	DeriveKey (Fixed, eeTarget)	
5	Nonce (Fixed)	
6	CheckMac (authKey, Nonce from Output 3, SHA (authKey 3)	
7	Nonce (Fixed)	
8	MAC (eeTarget, TempKey)	
9		Nonce (Fixed)
10		GenDig (hostKey)
11		CheckMac (Input 7, TempKey, and Output 8)

4.5 Separate Key Authentication (Verify), No MAC

Step	To Client ATSHA204	To Host ATSHA204
1	Nonce (Message)	
2	Verify (authKey, Message from Step 1, ECDSA (authKey 1)	
3	Nonce (Fixed)	
4	DeriveKey (Fixed, eeTarget)	
5	Nonce (Fixed)	
6	MAC (eeTarget, TempKey)	

4.6 Roll, Required Random Nonce

ReqRandom for eeTarget should be one. No MAC, no authorization.

Step	To Client ATSHA204	To Host ATSHA204
1	Nonce (Random)	
2	DeriveKey (TempKey, eeTarget)	
3	Nonce (Fixed)	
4	MAC (eeTarget, TempKey)	
5		Nonce (Fixed, SHA 1)
6		GenDig (hostKey)
7		CheckMac (Input 3, TempKey, and Output 4)

5. Data Validation

Slot0 contains data to be validated. slotConfig.isSecret is zero and slotConfig.writeConfig is 0000 (but could actually have any value). eeKey is some other slot which can have any appropriate key configuration value. “eeKey” refers to the updated value stored in the Client EEPRO.

5.1 MAC — External SHA and Public Data

Step	To Client ATSHA204	To Host ATSHA204
1	Nonce (Fixed)	
2	GenDig (slot0)	
3	MAC (eeKey, TempKey)	
4		Nonce (Fixed, SHA(Step 1, slot0))
5		CheckMac (eeKey, TempKey, Output 3)

5.2 MAC — External SHA and Validate OTP Values

This is the use case which prevents a man-in-the-middle from appearing to change the value of the OTP bits; otherwise, it’s identical to use case MAC — External SHA and Public Data.

Step	To Client ATSHA204	To Host ATSHA204
1	Nonce (Fixed)	
2	GenDig (OTP)	
3	MAC (eeKey, TempKey)	
4		Nonce (Fixed, SHA(Step 1, OTP))
5		CheckMac (eeKey, TempKey, Output 3)

5.3 MAC — External SHA and Possibly Secret Data

Note: This use case is essentially identical to that of validating data in multiple slots at the same time. Just run GenDig(Slot 1) (and/or Slot 2…) at some time prior to MAC and CheckMac.

Step	To Client ATSHA204	To Host ATSHA204
1	Nonce (Fixed)	
2	GenDig (slot0)	
3	GenDig (eeKey)	
4	MAC (TempKey, Fixed)	
5		Nonce (Fixed, Step 1)
6		GenDig (slot0)
7		GenDig (eeKey)
8		CheckMac (TempKey, Fixed Step 4, Output 4)

5.4 Single Use

Same as MAC — External SHA and Validate OTP Values, but the eeKey is Single Use and *must* be in Slot 0 to Slot 7, slotConfig.singleUse is one. The entire sequence can be run nine times (the ninth iteration should fail), or UseFlag can be initialized to 0x01; the sequence can be run two times (the second should fail).

5.5 Limited Use

Same as MAC — External SHA and Validate OTP Values, but the eeKey is Limited Use and *must* be in slot 15, and slotConfig.singleUse is one. The entire sequence can be run 129 times (the 129th sequence should fail), or LastKeyUse can be initialized to 0xFF FF… FE and the sequence can be run two times (the second sequence should fail).

5.6 HMAC, External SHA, and Public Data

Multiple slot validation is also possible with this sequence with the exception that the input to Nonce in Step 4 is SHA(SHA(Step 1, slot0), slot1) and so on.

Step	To Client ATSHA204	System Software
1	Nonce (Fixed)	
2	GenDig (slot0)	
3	HMAC (eeKey,TempKey)	
4		Compute Nonce as in Step 1.
5		Compute TempKey using output of Step 4 and slot0 value.
6		Compute SHA using eeKey and TempKey. Compare to Output 3.

5.7 Validate Data in Slot0 of Client Using Random Nonces on Both Sides

Expected that Slot0 is public on the Client; therefore no Read restrictions. There may or may not be Write restrictions. *No software SHA on Host.* Slot0 used as Scratch Register on Host *only* and should be free R/W on the Host.

Step	To Client ATSHA204	To Host ATSHA204
1		Random()
2	Read Slot N	
3	Nonce (Random, Output f/Step 1)	
4	GenDig (Slot N)	
5	Mac (eeKey)	
6		Write (Output from Step 2 into slot N)
7		Nonce (Fixed, Rand from Step 3)
8		Nonce (Calculation, Rand from Step 1)
9		GenDig (Slot N)
10		CheckMac (eeKey, Output 5)

6. Reads and Writes

It can reasonably be expected that most systems will include at least one combination of Read (Always, Never, and Encrypt) and Write (Always, Never, and Encrypt). Some potential use cases are described below. The sequences below describe encrypted Read-only and encrypted Write-only; combining these with the obvious cases is left to the tester.

Table 6-1. Potential Read and Write Use Cases

Read	Write	Use Case
Always	Always	Replacement for Serial EEPROM.
Always	Never	Model numbers, calibration, etc.
Always	Encrypt	ePurse
Never	Always	Key that can be destroyed, see below.
Never	Never	The most secure kind of key.
Never	Encrypt	Standard key that can be updated if you know the current value.
Encrypt	Always	Not commonly used.
Encrypt	Never	Confidential factory data.
Encrypt	Encrypt	Typical configuration for confidential field data and ePurse.

6.1 Encrypted Read

Data is to be read from SlotX. slotConfig.readKey for that slot should be eeKey. There is no support for encrypted reads in the Host chip.

Step	To Client ATSHA204	To Host ATSHA204
1	Nonce (Random)	
2	GenDig (eeKey)	
3	Read	

6.2 Encrypted Write

Data is to be written to SlotX, slotConfig.writeKey for that slot should be eeKey. There is no support for encrypted Writes in the host chip.

Step	To Client ATSHA204	To Host ATSHA204
1	Nonce (Random)	
2	GenDig (eeKey)	
3	Write	

6.3 Secure Personalization

Similar to Encrypted Write, except that the encryption should always be permitted regardless of the slot number passed to GenDig.

Step	To Client ATSHA204	To Host ATSHA204
1	Nonce (Random)	
2	GenDig (eeKey)	
3	Write	

7. Password Checks

This capability provides a way to securely check that the system knows a secret value (i.e. password) without forcing the output of the ATSHA204 device to be random.

Two slots are used in this sequence:

- Slot P (the “password”)
- Slot S (the “secret”)

Slot P should be even (i.e. the LSP of the KeyID/SlotNum should be zero) and Slot S should equal Slot P+1. In the use case — Section 7.2, “Use Password Directly as Key”, Slot Pand Slot S are identical and should be odd.

7.1 Password Map

If the system proves knowledge of the password P, then the value of the secret S is copied into TempKey for subsequent use by the MAC command. Essentially, this maps a password (that might have poor entropy) into a secret (that could have high entropy). This is the mechanism that would be used to decrypt files stored on a flash drive (32 byte blob stored with file is combined with S to generate the AES key). The values of slot config are critical:

Table 7-1. Slot P

Config Field	Value	Notes
ReadKey	Non-zero	
CheckOnly	1	Does not permit exhaustive password attack on a stolen drive.
SingleUse	0	Need to use this slot many times with CheckMac.
EncryptRead	0	Reads never permitted.
IsSecret	1	Reads never permitted.
WriteKey	ParentKey	Points to key that can be used to write a recovery value into Slot P.
WriteConfig	0100	Encrypted Write okay; DeriveKey not permitted.

Table 7-2. Slot S

Config Field	Value	Notes
ReadKey	0	Enables special mode.
CheckOnly	1	Does not permit use of Slot S with MAC (redundant to SingleUse).
SingleUse	1	Set UseFlag to zero. This key can cannot be used with *any* command other than CheckMac copy operation.
EncryptRead	0	Reads never permitted.
IsSecret	1	Reads never permitted.
WriteKey	X	
WriteConfig	1000	Writes never permitted.

This capability is similar to the authentication mechanism of the TPM which requires knowledge of a secret (authorization value) in order to use a key.

Step	To Client ATSHA204	System Software
1	Nonce (Random)	
2		Compute Nonce value from output of Step 1.
3		Combine the result of Step 2 with Password using SHA.
4	CheckMac (P, TempKey, and Step 3)	
3	MAC (TempKey, and Fixed)	

7.2 Use Password Directly as Key

Password is used directly as the secret for the subsequent MAC calculation. This mode permits interoperability as the capability can be duplicated by external (presumably secure) software without the knowledge of a secret key. This is an alternate mechanism that would be used to decrypt files stored on a flash drive (32 byte blob stored with file is combined with P to generate the AES key). As with use case, Password Map, slot config is critical.

Table 7-3. Slot P = S

Config Field	Value	Notes
ReadKey	0	Enables special mode.
CheckOnly	1	Don’t permit use with MAC.
SingleUse	0	Need to use this slot many times with CheckMac.
EncryptRead	0	Reads never permitted.
IsSecret	1	Reads never permitted.
WriteKey	ParentKey	Points to key that can be used to write a recovery value into Slot P.
WriteConfig	1100	Encrypted Write okay; DeriveKey not permitted.

This model may be less secure than the Password Map use case above since there are additional offline exhaustive password attack opportunities.

Step	To Client ATSHA204	System Software
1	Nonce (Random)	
2		Compute Nonce value from output of Step 1.
3		Combine result of Step 2 with Password using SHA.
4	CheckMac (P, TempKey, and Step 3)	
3	MAC (TempKey and Fixed)	

8. Write Lock

The OEM programs the configuration section, locks it, and then programs all slots with the appropriate combination of OEM secrets and fixed data, then locks the data section. The Subcontractor then overwrites those slots for which it has the appropriate authority.

All slots to be write-locked should have WriteKey point to Slot P, and have WriteConfig set to Encrypted.

8.1 Single Use Feature — Eight or Fewer Write-locked Items Required

Note: In practice, expecting to use all eight uses productively is probably low yield. Atmel recommends no more than six uses. The example assumes Slot P is zero.

Table 8-1. Slot P (Parent)

Config Field	Value	Notes
ReadKey	Non-zero	
CheckOnly	0	Need to use this for GenDig.
SingleUse	1	See notes, should be zero in some cases.
EncryptRead	0	Reads never permitted.
IsSecret	1	Reads never permitted.
WriteKey	P	Best default value.
WriteConfig	Never	No need to ever write this key at the subcontractor.

Step	To ATSHA204	Notes
1	Nonce (Random)	
2	GenDig (P)	
3	Write (Encrypted, Slot 1)	
4	…	Repeat Steps 1 – 3 five more times on Slots 2 – 6.
5	MAC (Junk, P)	
6	MAC (Junk, P)	This finishes off useFlag.

8.2 Single Use Feature — Arbitrary Number of Write-locked Items Required

GrandParent is programmed to a random number by the OEM prior to locking the data slots. The example assumes P is zero and GP is one.

Table 8-2. Slot P (Parent)

Config Field	Value	Notes
ReadKey	Non-zero	
CheckOnly	0	Need to use this for GenDig.
SingleUse	0	Must be zero since use is 14 times.
EncryptRead	0	Reads never permitted.
IsSecret	1	Reads never permitted.
WriteKey	GP	
WriteConfig	Parent	DeriveKey using parent (create); no auth.

Table 8-3. Slot GP (Parent)

Config Field	Value	Notes
ReadKey	Non-zero	
CheckOnly	0	Need to use this for DeriveKey.
SingleUse	1	Initialize UseFlag(GP) to leave two uses remaining.
EncryptRead	0	Reads never permitted.
IsSecret	1	Reads never permitted.
WriteKey	GP	Best default value.
WriteConfig	Never	No need to ever write this key at the subcontractor.

Step	To ATSHA204	Notes
1	Nonce (Random)	
2	GenDig (P)	
3	Write (Encrypted, Slot 2)	
4	…	Repeat Steps 1 – 3 fourteen more times on Slots 3 – 15.
5	Nonce (Random)	
6	DeriveKey (P)	Writes really random value into Slot P.
7	MAC (Junk, GP)	
8	MAC (Junk, GP)	This finishes off useFlag.

8.3 Random Keyset Strategy — Arbitrary Number of Items to be Write-locked

Slot P is configured as specified below.

Table 8-4. Slot P (Parent)

Config Field	Value	Notes
ReadKey	Non-zero	
CheckOnly	0	Need to use this for GenDig.
SingleUse	0	
EncryptRead	0	Reads never permitted.
IsSecret	1	Reads never permitted.
WriteKey	P	Important to prevent writes in the field.
WriteConfig	Roll	DeriveKey using target (roll); no authorization.

Step	To ATSHA204	Notes
1	Nonce (Random)	
2	GenDig (P)	
3	Write (Encrypted, Slot 0)	
4	…	Repeat Steps 1 – 3 fifteen more times on Slots 1 – 15.
5	Nonce (Random)	
6	DeriveKey (P)	Roll Slot P to random value.

9. Generate a Random Key

When a system needs to encrypt data such that it can only be decrypted by this same system, then a truly random key, not even known to the local operator, is useful.

This mechanism uses two slots:

- **Slot K (The Random Key)**:
  Slot K should be non-readable, but should require an authorized DeriveKey(Roll) command to be changed. Without the authorization, the system would be susceptible to a DOS attack.
- **Slot P (The Parent of that Key)**:
  P should be writeable with knowledge of the current P value.

9.1 Secrecy of the Key — Disassociation Between Two Key Generation Operations

The secrecy of the key largely depends on the disassociation between two key generation operations. The example assumes that the OEM writes a random value into Slot K, and that multiple users (e.g. the IT department, then the end user) could perform the Roll operation.

Step	To Client ATSHA204	Notes
1	Lock (Config)	
2	Write (PlainText, P)	Default parent value, published.
3	Lock (Data)	
4	Nonce (Random)	
5	GenDig (Transport)	Value of transport key does not need to be known by anyone.
6	DeriveKey (K)	Slot K is written to Rand + Transport + OldK.
		… steps below done by end customer...
7	Nonce (Random)	
8	Write (Encrypted, P, newPvalue)	Do not leave published value here.
9	Nonce (Random)	
10	DeriveKey (K)	Could repeat Steps 9 and 10 as desired.

10. Third Party Enablement

In some cases, there may be a concern that pre-programmed parts may be stolen during shipment or from a subcontractor inventory, thereby enabling fraudulent production of components that appear to be ‘authorized’ by the OEM.

Alternatively, it may be viewed that two trusted parties are required to generate a particular secret:

- One that performs the usual secure personalization, and
- A separate third party that modifies a secret key in the device with the addition of a subsequent secret.

This capability is implemented by having Atmel (or the OEM) securely personalize the part including storing a secret in the Client. The Data zone is locked prior to shipment to the third party.

The third party is given a separate secret that will be required to be known by the device in order to enable proper operation of the Client in the field. As part of the Client component test, the subcon generates and sends the necessary sequence to the chip to complete the personalization.

In the flows below,:

- eeKey refers to the updated value stored in the Client EEPROM.
- Subcon refers to the value known only by the subcon.
- Root refers to the original OEM secret value stored in the Client prior to updating and permanently in the Host.

10.1 Simplest Scheme

The simplest scheme involves generating a fixed, updated secret into an EEPROM slot on the Client and using a Host device to generate the matching secret in TempKey.

Table 10-1. Third Party Enablement

Step	To Client ATSHA204	Notes
1	Nonce (Fixed = K)	
2	DeriveKey (eeKey, Mac (Subcon))	Roll operation: eeKey<-SHA(Root…K)
3	MAC (eeKey)	Optional fixed challenge-response for testing.

Table 10-2. End Use Authentication

Step	To Host ATSHA204	Notes
1	Random( )	Send to Client as challenge, save for step 4.
2	Nonce (Fixed = K)	
3	GenDig (Root)	TempKey<-SHA(Root…K).
4	CheckMac (TK, params 1)	

10.2 Client Key — EEPROM

Same as the simplest scheme described above, but this time the Client key is generated in EEPROM in the Host for faster subsequent authentication.

Table 10-3. Third Party Enablement

Step	To Client ATSHA204	Notes
1	Nonce (Fixed = K)	
2	DeriveKey (eeKey, MAC(Subcon))	Roll Operation: eeKey<-sha(Root…K).
3	MAC (eeKey)	Optional fixed challenge-response for testing.

Table 10-4. End Use Authentication

Step	To Host ATSHA204	Notes
1	Nonce (Fixed=K)	
2	DeriveKey (Root)	Derive: eeKey<-sha(Root…K)
3	Random( )	Send to Client as challenge, save for Step 4.
4	CheckMac (eeKey and params 3)	

10.3 Client Key — Diversified

Same as Simplest Scheme, but this time the Client key is diversified. Applies to Section 10.2, “Client Key — EEPROM” as well by replacing K with SN.

Table 10-5. Third Party Enablement

Step	To Client ATSHA204	Notes
1	Nonce (Fixed = SN)	3rd party reads SN from Configuration zone.
2	DeriveKey (eeKey, MAC(Subcon))	Roll Operation: eeKey<-sha(Root…SN).
3	MAC (eeKey)	Optional fixed challenge-response for testing.

Table 10-6. End Use Authentication

Step	To Host ATSHA204	Notes
1	Random( )	Send to Client as challenge; save for Step 4.
2	Nonce (Fixed=SN)	Read SN from Client prior to Step 1.
3	GenDig (Root)	TempKey<-sha(Root…SN).
4	CheckMac (TK and param 1)	

11. Revision History

Doc. Rev.	Date	Comments
8849A	04/2013	Initial document release.

Atmel Corporation 1600 Technology Drive, San Jose, CA 95110 USA **T:** (+1)(408) 441.0311 **F:** (+1)(408) 436.4200 | **www.atmel.com**