# Application Note How to Create and Program Authentication Keys into TI Battery Fuel Gauges



Kipp Hayes

#### ABSTRACT

Most Texas Instruments battery fuel gauge products support authentication for anti-counterfeit protection of battery packs. This application note describes the process of creating a SHA or ECC key using BQKEYPACKAGER, programming the selected key onto a gauge using BQKEYPROGRAMMER, as well as guidance on programming the key in a production environment.

The purpose of these two tools being standalone is to allow an OEM to encrypt authentication keys in a file and send to a packmaker. The packmaker can then use this file as the data source for BQKEYPROGRAMMER to program the key onto a gauge. This way the key is not visible to the packmaker.

- BQKEYPACKAGER
- BQKEYPROGRAMMER
- BQSTUDIO

### **Table of Contents**

1 Generating a Key File with BQKEYPACKAGER	2
1.1 SHA1 Process	3
1.2 SHA256 Process	4
1.3 ECC Process	6
2 Programming a Key Into the Fuel Gauge With BQKEYPROGRAMMER	8
3 Programming Kevs in Production	
4 Summary	10
5 References	10

### Trademarks

All trademarks are the property of their respective owners.



### 1 Generating a Key File with BQKEYPACKAGER

BQKEYPACKAGER is intended only for an OEM to allow packaging of the authentication key. The authentication key is entered along with a password that is used in an algorithm to encrypt the file. After testing using BQKEYPROGRAMMER or BQSTUDIO, an OEM can then send the created file and password to a packmaker. Texas Instruments battery gauges have 3 methods of authentication, SHA1,SHA256, and ECC.

The output of the BQKEYPACKAGER tool is a .bqk file. This file is an encrypted output of the key generated using the tool. For the packaged .bqk file there are 2 options for output. A single file that encompasses the entire key or 2 separate files that each have one half of the key, these being part F and part C. When programming the key in production, the key can be programmed one half at a time or both halves at the same time. This allows for separate halves of the key to be programed at separate times and locations during production. For example, if you have a manufacturing flow with 2 manufacturers that the battery pack goes through. Each manufacturer can be given one half of the key to program into the gauge. This increases the security of the key as no single manufacturer has the entire key, only the OEM or designer has the key.



Texas Instruments is not responsible for the security of any key packaged or programmed using BQKEYPACKAGER or BQKEYPROGRAMMER respectively. The user is responsible to make sure the key is kept secure and to limit the number of parties that have access to any part of the key.



Figure 1-1. Split Key Production Example

#### 1.1 SHA1 Process

Step 1: Download and install BQKEYPACKAGER.

Step 2: Open BQKEYPACKAGER and select the correct authentication scheme and corresponding device. This example selected SHA1 Key BQ40Z80.

Step 3: Input the SHA1 authentication Key F and C parts respectively, these are each one half of the final key. For this example, Key F is AAAA1111BBBB2222CCCC3333DDDD4444EEEE5555, and Key C is *FFFF666667777888899990000AAAA1111BBBB2222*.

Step 4: Enter a password for the generated .bqk file and any comments to display in BQKEYPROGRAMMER. For this example, the password is *example1234567890*, and the comment is *example*.

Step 5: Select Create bqKey file then choose output destination and file name.

👼 bqKeyPackager			83 <del>7 - 5</del> 5		$\times$
File Help					
🗖 bqKey Packager 🕸					
bqKey Packager					
Package and encrypt keys for use	with bqKe	yProgrammer			
This tool will package keys securely Send both bqKey file and password See top menu "Help" for more info	r in a file and d to pack m prmation.	d encrypt the file based on a	entered password	. This file	is used
Select device with type of keys SHA1 Key bq40z80 ~	to packag	e for bqProgrammer			
Enter SHA1 Secure Memory Ke	ys to Prog	ram. Most significant byt	e first.		
Enter SHA1 Authentication Key F	B2222CCCC	3333DDD4444EEEE5555			
Enter SHA1 Authentication Key C	888899990	000AAAA1111BBBB2222			
Enter password to encrypt the	bqKeyFile	with a comment			
Enter password ten or more chara Enter Comment to display in bqKe	<mark>cters</mark> yProgramm	example1234567890 ner.			
example					
🖋 Create bqKey File					
<					>
			-	TEXAS INS	TRUMENT

Figure 1-2. SHA1 Example with BQ40Z80



### 1.1.1 How to Calculate SHA1 Key

Start by calculating the SHA-1 hash of the 20-byte Key F sent to the gauge, in reverse byte order of what was sent. Key F is the last 8-bytes. Key C uses same procedure. The final key is Key C appended to Key F.

Example:

If 2309BDC0A9F86B69111CA850B530339111000C47 (little endian) is for KeyF

SHA-1 hash input to find keyF: 470C0011913330B550A81C11696BF8A9C0BD0923 (big endian)

SHA-1 hash output: 8877626BD64ABC4843E43F1E42C5413DB9EDBBA2

If 330C0014913530B550A81D10696BF8A9C7BD0613 (little endian) is for KeyC

SHA-1 hash input to find keyC: 1306BDC7A9F86B69101DA850B530359114000C33 (big endian)

SHA-1 hash output: 1B8689D48F55F5B5F81D30E012DFB96B7440433C

KeyF = 42C5413DB9EDBBA2, KeyC = 12DFB96B7440433C

SHA-1 hash input for all 0's challenge:

Response: 1625A385F73436792FD693D727BAC6EE47291D0B

Second hash is the response of the first appended to the key:

KeyF + KeyC + Response from first hash = 42C5413DB9EDBBA212DFB96B7440433C1625A385F73436792FD693D727BAC6EE47291D0B

Response: B0BEF440CAC492FAF09604A46C2C0996512331AF

This response needs to correspond to the data received from the gauge after using an all 0's challenge.

#### 1.2 SHA256 Process

4

For SHA256, there are 2 ways of generating a key, either inputting 2 challenges and the key can be generated from the challenges, or inputting a pre-generated key.

Step 1: Download and install BQKEYPACKAGER.

Step 2: Open BQKEYPACKAGER and select the correct authentication scheme and corresponding device. For this example we selected SHA256 Key BQ9035.

Step 3: Input either the 2 challenges or single per-generated key. For this example the 2 challenges input are 123456789012345678901234567890123456789012345678901234 and 09876543210987654321098765432109876543210987. The pre-generated key in this example is AAAA1111BBBB2222CCCC3333DDDD4444EEEE5555FFFF666667777888899990000.

Step 4: Enter a password for the generated .bqk file and any comments to display in BQKEYPROGRAMMER. For these examples, the password is *example1234567890*, and the comment is *example*.

Step 5: Select *Create bqKey file* and choose output destination and file name. If using the SHA256 Challenge method, after creating the file, there is a pop up window with the full generated key. In this example the output key is *CA1194A558362B5FA6E7887DA7B41EC65481031C133249274853B0559D887BA3*.

bqKeyPackager					$\times$
ile Help					
bqKey Packager ⊠					
oqKey Packager					
ackage and encrypt keys for use	with bqKe	yProgrammer			
This tool will package keys securely	in a file and	l encrypt the file based on entered pa	issword	l. T <mark>his f</mark> ile	is use
Send both bqKey file and password See top menu "Help" for more info	l to pack m rmation.	aker after testing.			
Select device with type of keys	to packag	e for bqProgrammer			
SHA256 Key bq9035 V					
Enter SHA256 Secure Memory k	eys to Pro	ogram. Most significant byte first.			
• Challenge (20-32 bytes/40-64 h	ex) 🔿 Key	(32 bytes/64 hex)			
Enter SHA256 Challenge 1	34567	89012345678901234567890123456	789012	34	
Enter SHA256 Challenge 2	87654	321098765432109876543210987654	432109	87	
Enter SHA256 Authentication Key					
Select Key Slot to Program	1 ~				
Enter password to encrypt the b	qKeyFile	with a comment			
Enter password ten or more charac	ters	example1234567890			
Enter Comment to display in bqKey	/Programm	er.			
example					

Figure 1-3. SHA256 Challenge Example with BQ9035



🥏 bqKeyPackager					$\times$
File Help					
🗖 bqKey Packager 🛛					- 0
bqKey Packager					
Package and encrypt keys for us	e with bqKey	yProgrammer			
This tool will package keys secure Send both bqKey file and passwo See top menu "Help" for more inf	ly in a file and rd to pack ma formation.	l encrypt the file based on entere aker after testing.	d password	l. This file	is used
Select device with type of key	s to nackage	e for baProgrammer			
SHA256 Key bq9035	/	i of by rogrammer			
Enter SHA256 Secure Memory Challenge (20-32 bytes/40-64) Enter SHA256 Challenge 1 Enter SHA256 Challenge 2 Enter SHA256 Authentication Key Select Key Slot to Program	Keys to Pro hex)  Key ( 33DDD	ogram. Most significant byte fi (32 bytes/64 hex) D4444EEEE5555FFFF6666677778	r <b>st.</b> 8889999900	od	
Enter password to encrypt the Enter password ten or more char Enter Comment to display in bqK example	bqKeyFile v acters eyProgramm	with a comment example1234567890 er.			
٢					>
			-	TEXAS INS	TRUMENTS

Figure 1-4. SHA256 Key Example with BQ9035

🥏 SHA2	256 Key	$\times$
i	Generated SHA256 Key : CA1194A558362B5FA6E7887DA7B41EC65481031C133249274853B0559D887BA3	
	ОК	

Figure 1-5. SHA256 Generated Key Output

### **1.3 ECC Process**

The ECC keys come in a pair - the private key and the matching public key. Each key is 163-bit in length.

Both keys are required to be programmed into the gauge to run ECC authentication. The private key, which is the secret of the ECC authentication, is split into two (called KeyF and KeyC). Each 1/2 key can be programmed by a different entity (for example, by TI, pack makers, or system makers). This split keys programming option

allows the OEM to be the only one knowing the actual value of the private key, while any programming entity only has partial knowledge of the private key.

Step 1: Download and install BQKEYPACKAGER.

Step 2: Open BQKEYPACKAGER and select the correct authentication scheme and corresponding device. For this example we selected ECC Key BQ40Z80.

Step 3: Input Authentication Key F + Key C, in this example the key is AAAA1111BBBB2222CCCC3333DDDD4444EEEE5555FF. Input the Public Key, in this example the key is FF66667777888899990000AAAA1111BBBB2222CCCC.

Step 4: Enter a password for the generated .bqk file and any comments to display in BQKEYPROGRAMMER. For these examples, the password is *example1234567890*, and the comment is *example*.

Step 5: Select Create bqKey file and choose output destination and file name.

뼫 bqKeyPackager					$\times$
File Help					
⊐ bqKey Packager 🛛					- 0
bqKey Packager					
Package and encrypt keys for use with	n bqKeyPı	rogrammer			
This tool will package keys securely in a Send both bqKey file and password to p See top menu "Help" for more informat	file and er pack make tion.	ncrypt the file based on entered er after testing.	l password	. This file i	s used
Select device with type of keys to p	ackage f	or bqProgrammer			
ECC Key bq40z80 ~					
Enter ECC Secure Memory Keys to Enter ECC Authentication Key F + Key C Enter ECC Public Key	Program 22CCCC3 3999000	Most significant byte first. 3333DDDD44444EEEE5555FF 0AAAA1111BBBB22222CCCC			
Enter password to encrypt the bqKe	eyFile wit	h a comment			
Enter password ten or more characters		example1234567890			
Enter Comment to display in bqKeyProg	grammer.				
example					
Create bqKey File					
<					>
			-	Texas Inst	RUMENT

Figure 1-6. ECC Example With BQ40z80



### 2 Programming a Key Into the Fuel Gauge With BQKEYPROGRAMMER

BQKEYPROGRAMMER programs the packaged information from the bqKey file supplied by an OEM into the secure one time programmable memory in a supported gauge. This tool requires a working EV2x00 communications adapter with drivers and a compatible gauge connected to the communication port of the adapter.

Step 1: Connect the EV2x00 to the computer.

Step 2: Connect the target gauge to the EV2x00 via the correct communication port for the device(SMBus,I2C,HDQ).

Step 3: Install and open BQKEYPROGRAMMER.

Step 4: Click Select File and select the .bqk file.

Step 5: Enter the password, in this example the password is "example12345678".

Step 6: Click Load File, if the comment appears in the comment box, then the file is successfully loaded.

Step 7: **Programming is one time only and can not be reversed.** Click *Program Key*, If the key is successfully programmed, then a green box needs to appear stating *Key Programmed and verified Passed* as shown in Figure 2-1.

🕶 bqKey Programmer	- 🗆 ×	
File Help		
📩 bqKeyProgrammer 🖾	□ [	3
Secure Authentication Key Programmer		
This will program and verify secure authentication keys prese	nt in encrypted bqKey file.	
Import bqKey File		
Select Key File	and enter password	
🥜 Select File File Name	C:\Users\a0503086\Documents\examplekey.bq	
Password	example12345678	
📀 Load File		
Comment in bqKey File		
this is an example	^	
	~	
Program Key		
W Program Key Programmed and verifie	d Passed	
🐺 Verify Key		
	👋 Texas Instrumen	ITS



#### Note

BQKEYPROGRAMMER programs the one time programmable memory. If communication is lost or the incorrect bqKey file is used, the device is not useable for authentication. Check the comments in the bqKey file to make sure the correct file is selected. Make sure the programmer is finished before disconnecting communications and power. A complete power on reset is needed after programming to lock memory.

One way to test if your authentication successfully works is to use BQSTUDIO.

- Step 1: Install and open BQSTUDIO.
- Step 2: Open the Authentication tab.
- Step 3: In the Gauge Authentication by Host box, input the Key then click Load Gauge Key.
- Step 4: Click Generate Random Challenge then Authenticate Gauge.

If a green check mark appears, then you have successfully verified authentication functionality on the gauge.

Gauge Authenticati	ion by Host	
Key (256 bits):	AAA111BBB222CCC333DDD444EEE555FFF666AAA111BBB222CCC333DDD444EEE5	
		👸 Load Gauge Key
Challenge:	B3D54372289832F66DA90F370841B72149EC21F6D58B04712E8BD9C068C047B3	🤤 Generate Random Challenge
Expected Digest:	0DE4B2500C855AB061BD83562589AAA4DD647EFDD4E67AFF5A6024DBED00A0B6	🐺 Authenticate Gauge
Gauge Digest:	0DE4B2500C855AB061BD83562589AAA4DD647EFDD4E67AFF5A6024DBED00A0B6	•

#### Figure 2-2. BQSTUDIO Key Verification Example

### **3 Programming Keys in Production**

For help with programming authentication in production, BQKEYPROGRAMMER has a *Generate FlashStream File* function that can export a FlashStream file containing the I2C commands to program authentication into the gauge. This FlashStream file can then be parsed into a production environment to program that specific authentication key or key half into the corresponding gauge. This FlashStream file has the raw decrypted authentication key in the file and is not secure.

#### Note

BQKEYPACKAGER and BQKEYPROGRAMMER are not intended to be directly used in a production environment and are not supported as such. Texas Instruments is not responsible for the security of any key packaged or programmed using BQKEYPACKAGER or BQKEYPROGRAMMER respectively. The user is responsible to make sure the key is kept secure and to limit the number of parties who have access to any part of the key.

# 4 Summary

In summary, this application note describes the process of creating a SHA or ECC key using BQKEYPACKAGER, programming the selected key onto a gauge using BQKEYPROGRAMMER, as well as guidance on programming the key in a production environment.

# **5** References

- 1. Texas Instruments, [FAQ] What is the procedure to program the SHA-1 secure key?
- 2. Texas Instruments, TI Fuel Gauge Authentication Key Packager and Programmer Tools, user's guide.

### IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATA SHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES "AS IS" AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS AND IMPLIED, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for skilled developers designing with TI products. You are solely responsible for (1) selecting the appropriate TI products for your application, (2) designing, validating and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, regulatory or other requirements.

These resources are subject to change without notice. TI grants you permission to use these resources only for development of an application that uses the TI products described in the resource. Other reproduction and display of these resources is prohibited. No license is granted to any other TI intellectual property right or to any third party intellectual property right. TI disclaims responsibility for, and you will fully indemnify TI and its representatives against, any claims, damages, costs, losses, and liabilities arising out of your use of these resources.

TI's products are provided subject to TI's Terms of Sale or other applicable terms available either on ti.com or provided in conjunction with such TI products. TI's provision of these resources does not expand or otherwise alter TI's applicable warranties or warranty disclaimers for TI products.

TI objects to and rejects any additional or different terms you may have proposed.

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265 Copyright © 2025, Texas Instruments Incorporated