How do I set CAPKs (Certification Authority Public Keys? (L3 certification purposes)

One of the common questions during L3 certification - a test case requires SDA/CDA/DDA offline authentication and it's not working. Why?

You probably don't have your keys loaded. Let's get into this.

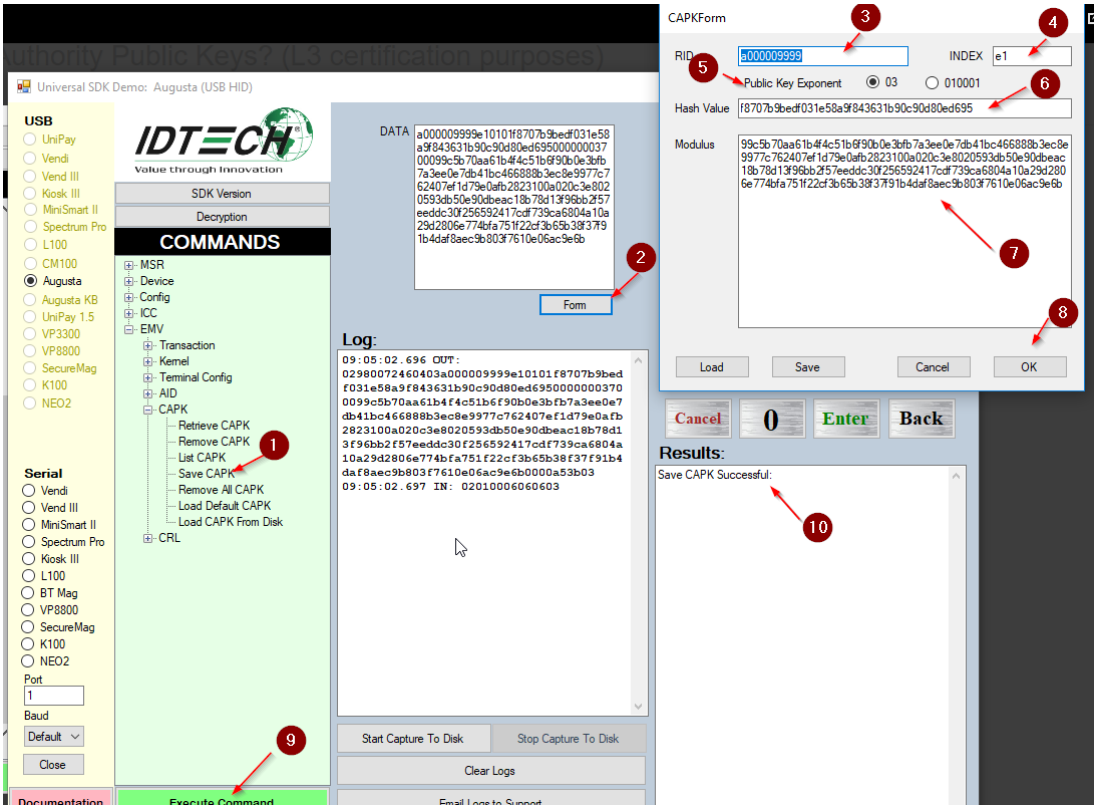
Note: This article's focus is on individuals who are certifying with the brands, but you can also loosely apply this to other use cases that require CAPKs to be loaded.

Edit: 11 Oct 2017 **: For the users of the C++ SDK, if you are getting errors with the length of the contactless CAPK modulus, flip the length bytes (i.e., use big-endian format, not little-endian). See the discussion further below.**

Saving a CAPK with the Universal Demo:

Look for the '**Save CAPK**' command, enter in the CAPK RID as well as the index.

Instructions on how to format the Data can be found further below.



You can grab the Universal Demo (most recent version) from here.

What do I need before I try to save a CAPK?

If you happen to be performing brand testing with a specific card - each test case lists which card is being used.

Often times, the cards are re-used among tests.

To pass these tests, some of them require SDA/CDA/DDA, which references a Public Key / Private Key pair.

The test will be using the private key on the card - you will need to load the respective public key onto the device that is being validated.

The below values in the 'Breaking the Data down' should be provided by the issuer/brand that requires the test.

Here's an example of how the data may be presented (heavily redacted) -

3.5 Index

Header	
Service Identifier	
Length of CAPK Modulus	
CAPK Algorithm Indicator	
Length of CAPK Exponent	
RID	
CAPK Index	
CAPK Modulus	
CAPK Exponent	
Hash Value	

Save CAPK command Data Breakdown (Sample):

Name:

a000009999e1

Data:

a000009999e10101f8707b9bedf031e58a9f843631b90c90d80ed695000000003700099c5b70aa61b4f4c51

b6f90b0e3bfb7a3ee0e7db41bc466888b3ec8e9977c762407ef1d79e0afb2823100a020c3e8020593db50e

90dbeac18b78d13f96bb2f57eeddc30f256592417cdf739ca6804a10a29d2806e774bfa751f22cf3b65b38f3

7f91b4daf8aec9b803f7610e06ac9e6b

Breaking the Data down (example AID):

AID RID (5 bytes)	a000009999
CA Index (1 byte)	e1
Hash Algorithm (1 byte)	01
Encryption Algorithm (1 byte)	01
Hash Value (20 bytes)	f8707b9bedf031e58a9f843631b90c90d80ed695
Public Key Exponent (4 bytes)	00 00 00 03
2 bytes Modulus Length (4 bytes) **NOTE**: The bytes need to be FLIPPED *IE 00 70 in this case, evaluating to 112 length*	70 00

Variable bytes Modulus (variable bytes length indicated prior)	99c5b70aa61b4f4c51b6f90b0e3bfb7a3ee0e7db41bc466888b3ec8e9977c762407ef1d79e0afb2823100a020c3e802059 3db50e90dbeac18b78d13f96bb2f57eeddc30f256592417cdf739ca6804a10a29d2806e774bfa751f22cf3b65b38f37f91b4 daf8aec9b803f7610e06ac9e6b

Example 1:

We were given this from AMEX to put in our device -

INDEX C8
Header '20' (not used)
Service Identifier '00 00 00 00' (not used)
Length of CAPK Modulus '90' (Modulus Length, interpret as 90 00)
CAPK Algorithm Indicator '01' (Hash Algorithm)
Length of CAPK Exponent '01'
RID 'A0 00 00 00 25' (AID Name)
CAPK Index C8 (CA Index)
CAPK Modulus (Variable Bytes Modulus)
BF0CFCED708FB6B048E3014336EA24AA007D7967B8AA4E613D26D015C4FE7805D9DB131CED0D2A8ED504C3B5CCD48C33199E5A5BF644DA043B54DBF60276F05B1750FAB39098C7511D04BABC649482DDCF7CC42C8C435BAB8DD0EB1A620C31111D1AAAF9AF6571EEBD4CF5A08496D57E7ABDBB5180E0A42DA869AB95FB620EFF2641C3702AF3BE0B0C138EAEF202E21D
CAPK Exponent '03' (Public Key exponent, would be viewed as 00 00 00 03)
Hash Value 33BD7A059FAB094939B90A8F35845C9DC779BD50

Let's piece this together:

Step 1: AID Name would be RID + CA index.

A0 00 00 00 25 + C8 = a000000025c8

Step 2: Rest of the body is formed by AID Name + CA Index (1 byte) + Hash Algorithm (1 byte) + Encryption Algorithm (1 byte) + Hash Value (20 bytes) + Public Key Exponent (4 bytes) + 2 bytes Modulus Length (4 bytes) + Variable bytes Modulus

a000000025 + c8 + 01 + 01 + 33BD7A059FAB094939B90A8F35845C9DC779BD50 + 00 00 00 03+ 9000
+ BF0CFCED708FB6B048E3014336EA24AA007D7967B8AA4E613D26D015C4FE7805D9DB131CED0D2A8ED504C3B5CCD48C33199E5A5BF644DA043B54DBF60276F05B1750FAB39098C7511D04BABC649482DDCF7CC42C8C435BAB8DD0EB1A620C31111D1AAAF9AF6571EEBD4CF5A08496D57E7ABDBB5180E0A42DA869AB95FB620EFF2641C3702AF3BE0B0C138EAEF202E21D

=

a000000025c8010133BD7A059FAB094939B90A8F35845C9DC779BD50000000039000BF0CFCED708FB6B048E3014336EA24AA007D7967B8AA4E613D26D015C4FE7805D9DB131CED0D2A8ED504C3B5CCD48C33199E5A5BF644DA043B54DBF60276F05B1750FAB39098C7511D04BABC649482DDCF7CC42C8C435BAB8DD0EB1A620C31111D1AAAF9AF6571EEBD4CF5A08496D57E7ABDBB5180E0A42DA869AB95FB620EFF2641C3702AF3BE0B0C138EAEF202E21D

Notes:

- Encryption Algorithm / Hash Algorithm are usually value 01
- If Hash value is not provided, you can calculate it as well (your acquirer should have provided this value). It is a SHA-1 hash of... see below.

How do I form the Hash Value? Concatinate the 4 data elements listed below.

AID RID + CA Index + Modulus + Public Key Exponent (hash only relevant bytes) In this example, I was able to re-calculated the SHA-1 hash successfully by removing the first three 00 bytes of the Public Key Exponent.

> ***Input for the hash :***
>
> ***A0 00 00 00 25 C8***
> ***BF0CFCED708FB6B048E3014336EA24AA007D7967B8AA4E613D26D015C4FE7805D9DB131CED0D2A8ED504C3B5CCD48C3***
> ***3199E5A5BF644DA043B54DBF60276F05B1750FAB39098C7511D04BABC649482DDCF7CC42C8C435BAB8DD0EB1A620C31***
> ***111D1AAAF9AF6571EEBD4CF5A08496D57E7ABDBB5180E0A42DA869AB95FB620EFF2641C3702AF3BE0B0C138EAEF202E***
> ***21D 03***
>
> ***Final hash:***
>
> ***33BD7A059FAB094939B90A8F35845C9DC779BD50***
>
> ***Use the tool at http://www.idtechproducts.com/hosted-files/tools/encryptiondecryptiontool.html to calculate SHA-1.*** *(The tool will automatically remove spaces and newlines from the data.)*

Another Example:

a000009999 +

e1+

99c5b70aa61b4f4c51b6f90b0e3bfb7a3ee0e7db41bc466888b3ec8e9977c762407ef1d79e0afb2823100a020c3e8020593db50e90dbeac18b78d13f96bb2f5
7eeddc30f256592417cdf739ca6804a10a29d2806e774bfa751f22cf3b65b38f37f91b4daf8aec9b803f7610e06ac9e6b +

03 (not 00 00 00 03)

**a000009999e199c5b70aa61b4f4c51b6f90b0e3bfb7a3ee0e7db41bc466888b3ec8e9977c762407ef1d79e0afb2823100a020c3e8020593db50e90dbeac1
8b78d13f96bb2f57eeddc30f256592417cdf739ca6804a10a29d2806e774bfa751f22cf3b65b38f37f91b4daf8aec9b803f7610e06ac9e6b03**

SHA-1 Hash on that value to obtain:

F8707B9BEDF031E58A9F843631B90C90D80ED695

CONTACTLESS CAPKS IN THE C++ / JAVA SDK

Known Issue: We have a minor edge case that has cropped up in recent days. This is being addressed. 11 Oct 2017

You can save this one in C++:
// Sample, note the **bolded** part
a000009999e50101ada2349afd118d55af782d37b64651af1ca61ee500000003 ***0080***
d4fdae94dedbecc6d20d38b01e91826dc6954338379917b2bb8a6b36b5d3b0c5eda60b337448baffebcc3abdba869e8dadec6c870110c42f5aab90a18f4f867f
72e3386ffc7e67e7ff94eba079e531b3cf329517e81c5dd9b3dc65db5f9043190be0be897e5fe48adf5d3bfa0585e076e554f26ec69814797f15669f4a255c13
(edited)

In C# demo / C# SDK, the above one won't actually save. But this will save in C#:

a000009999e50101ada2349afd118d55af782d37b64651af1ca61ee500000003 ***8000***
d4fdae94dedbecc6d20d38b01e91826dc6954338379917b2bb8a6b36b5d3b0c5eda60b337448baffebcc3abdba869e8dadec6c870110c42f5aab90a18f4f867f
72e3386ffc7e67e7ff94eba079e531b3cf329517e81c5dd9b3dc65db5f9043190be0be897e5fe48adf5d3bfa0585e076e554f26ec69814797f15669f4a255c13

In short, for your existing CAPK files you actually already loaded in, if we want to make them work with the original format

Swap the length bytes (i.e., use big-endian format in your code) and they should save.

Original created AID from C# demo
a000000004fa01017f5acbb96b589f74cb959ed1c35bdb965c3f410600010001 *00f8*
a4203e0c7beb27097b63c103c19fdcda671aea7f813065756f3b9b81810cbd4bc4dec548fbf1f3cdae51f847235cbf2c8badd8aca7c93bea3d44e80ed6a7b70e
29622619db420accce07e1dd4e6c354f359fbdc9c5b70813926f77d827e52b19daf09bfae5274438bb8f61d17753c9ec0a8efa3b7e46f02692160d2653cdbcc7
1b7d48bd37968316eb444f6504b9421b7dd3035a2c117d8b1f76a8975440da9563618102397b881cef8ada7689edface32482a2dffed656e7f951db841da783
68c6293bfc1053a86a845bfa6578e4b69f100b42b558fde1aecec6d250741bc783aa8a68a4261e7bb9246b10587a498d68dd955ce8b2b2433

Edited, the one you should use for C++ and moving on forwards
a000000004fa01017f5acbb96b589f74cb959ed1c35bdb965c3f410600010001 *f800*
a4203e0c7beb27097b63c103c19fdcda671aea7f813065756f3b9b81810cbd4bc4dec548fbf1f3cdae51f847235cbf2c8badd8aca7c93bea3d44e80ed6a7b70e
29622619db420accce07e1dd4e6c354f359fbdc9c5b70813926f77d827e52b19daf09bfae5274438bb8f61d17753c9ec0a8efa3b7e46f02692160d2653cdbcc7
1b7d48bd37968316eb444f6504b9421b7dd3035a2c117d8b1f76a8975440da9563618102397b881cef8ada7689edface32482a2dffed656e7f951db841da783
68c6293bfc1053a86a845bfa6578e4b69f100b42b558fde1aecec6d250741bc783aa8a68a4261e7bb9246b10587a498d68dd955ce8b2b2433

Questions, comments, concerns, want more examples? Leave a comment, and / or email me at jasonc@idtechproducts.com