Commonly Accepted Keys and CSPs Initiative

International Cryptographic Module Conference (ICMC)
Friday November 6th, 2015

© CGI Group Inc. 2015

Experience the commitment®

Introduction

Ryan Thomas
FIPS 140-2 Program Manager
CGI Global IT Security Labs
ry.thomas@cgi.com

Located in Ottawa, Canada

CGI

A little bit about CGI Global Labs

- Common Criteria evaluation laboratory since 1999
- Accredited Cryptographic and Security Testing (CST) Laboratory under NVLAP since May 2011
- Successfully completed just under 100 FIPS 140-2 module validations in that time

CGI

Common Keys and CSPs Initiative

CGI

Experience the commitment®

Background - What's this all about?

- FIPS 140-2 requires vendor Security Policies to list all Keys and Critical Security Parameters (CSPs) employed by their cryptographic modules

- AS.14.08, Appendix C and IG 14.1 require:

  - Sufficient detail on type of algorithm used by Key/CSP
  - A mapping of each Key/CSP to an Approved service
  - For each service the type of operator access to the Key/CSP

CGI

Problem Definition

- For Vendors: I know my products, I understand security. I need help listing Keys/CSPs in the way the CMVP want them!

- For Labs: We understand the requirements, we review many different types of vendor submissions that meet these requirements. I wish there was an Approved template we could use.

- For CMVP reviewers: Every Security Policy we see is different, even frequently used and common CSPs/Keys are not consistently documented. This results in more time spent during the review process!

CGI

The Idea … Make life simpler

1. Develop a Keys and CSPs Table template that addresses the requirements
2. Ask CMVP to review and provide guidance
3. Use this template for some of the more complex algorithms functions
4. Repeat Step #2
5. Distribute them and ask for additional feedback

CGI

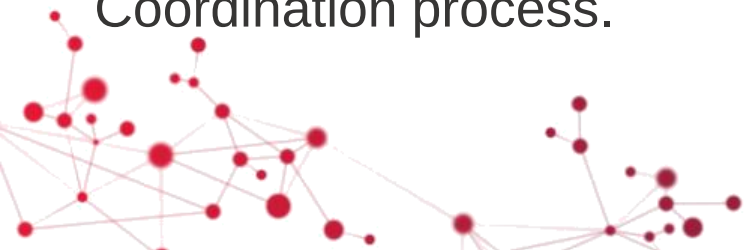
The Benefits

- For Vendors: Leverage the publically available pre-populated list of Keys/CSPs and customize to your implementation – save time and effort

- For Labs: Clear, consistent checklist vetted by CMVP – save time and effort. Gain a better understanding of what the CMVP are looking for from us in this Table

- For CMVP reviewers: Conformant table inputs with sufficient details. Less time to review = less comments for the vendor and lab!

Hopefully, this initiative will lead to improved Security Policy consistency and increase efficiency during the lab documentation review and CMVP Coordination process.



CGI

Inputs Used

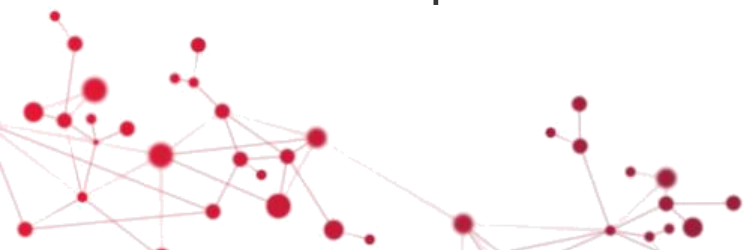
- NIST Special Publications
- IETF RFCs
- FIPS PUBS
- FIPS 140-2 Implementation Guidance
- Recently validated module Security Policies on CMVP website
- CMVP feedback
- CMVP's 2011 Security Policy template

CGI

Categorizing the Module Types

Obviously the type of module and module's cryptographic boundary will determine quite a bit when addressing Keys and CSPs

- Are we dealing with a Software or Hardware module?
- A Hardware module can handle many types of keys …
  - Persistent and Ephemeral (temporary/session keys) – more on this later
- A Software module such as cryptographic library likely will not store keys …
  - Keys likely passed to the module by application/calling process
  - Module operates with a reference to the key/CSP in memory



CGI

The Table - Template and Columns

Experience the commitment®

Keys and CSPs - Template

Key/CSP Name	Key/CSP Type	Generation/Input	Output	Storage	Zeroization	Use
AES key	AES 128-, 192-, or 256-bit key CTR and ECB modes	Input via API in plaintext	Exits in plaintext via API	Ephemeral. Generated each time a call is made	Reboot operating system; API call; Cycle host power	Encryption/ Decryption [Service 1] R/X; [Service 4] R; and [Service 8] R.

CGI

Column #1 – “Key/CSP Name”

Objective: list the type of key or CSP used by the cryptographic module.

To avoid confusion wherever possible it is recommended that the name of the key/CSP be consistent with a recognized industry standard such as ISO, IETF RFC or NIST Special Publication.

- Instead of “TLS key 1” “TLS key 2” it should be “TLS Authentication Key” and “TLS Session Key”

CGI

Column #2 – “Key/CSP Type”

Objective: Provide additional information about the CSP.

Where applicable this column shall include the type of key/CSP, algorithm(s) (including reference to FIPS or NIST SP) and size.

All modes for the algorithm shall also be listed.

- If you implement AES – include the key sizes (128/192/256) and the modes supported (ECB, CBC and CFB)

CGI

Columns #3 – “Generation/Input”

Objective: Specify how and when the CSP is generated, derived or enters the module.

Specify the function or technique responsible for generation or derivation.
Example: SP 800-90A DRBG or SP 800-133 PBKDFv2

If the CSP is entered, specify if the CSP is entered electronically or manually and if it is encrypted or plaintext form.

If ephemeral, specify conditions upon which it is generated
(Example: key is generated each time a call is made to the module).

CGI

Column #4 – “Output”

Objective: to specify if the CSP can be output from the cryptographic module.

If the CSP can be output, the column shall specify how it can be output and if it is encrypted or plaintext form.

CGI

Column #4 – “Storage”

Objective: to specify where the CSP is stored during the operation of the cryptographic module.

The location and type of storage shall be explicitly listed. (Example: FLASH/DRAM)

It shall also state if the CSP is persistent, ephemeral or hardcoded and if it is stored encrypted or in plaintext form.

The column shall specify if only a pointer or reference to the CSP is stored or the actual CSP.

CGI

Column #5 – “Use”

Objective: to provide information on how the key is used during cryptographic module operation.

It is important that each CSP can be directly mapped to an Approved service that the cryptographic module performs.

An indicator or unique identifier such as [Service 1] or [1] etc. can be used in order to facilitate the mapping of the Approved services Table listed in the Security Policy to each CSP listed in the Keys and CSPs Table.

The type of access for each CSP shall also be listed in either the Approved services Table or the Keys and CSPs Table. Example – read, write and execute

CGI

Columns #6 – “Zeroization”

Objective: to provide details on how the CSP shall be zeroized. All possible zeroization techniques for the CSP shall be listed.

- Zeroization command could be used to overwrite
- Power-cycle for ephemeral
- Procedure zeroization (multiple steps by CO to overwrite)

CGI

Example listings .. (Still draft)

Experience the commitment®

Warning - Your Implementation may vary

- The initial objective is to provide vendors/labs with a good starting point
- I recognize that depending on the type of module these may differ
- You can take these inputs and customize them for your products

NIST SP 800-90A DRBG

- Low hanging fruit …
- but there were still inconsistencies in the Security Policies we reviewed

CGI

NIST SP 800-90A Hash-based DRBG

Key/CSP Name	Key/CSP Type	Generation/Input	Output	Storage	Zeroization	Use
DRBG Seed	Pseudorandom string. 384-bits.	Generated using the DRBG derivation function. Includes entropy input from entropy source	Never exits the module	Plaintext in volatile memory	Power cycle.	Input that determines the internal state of the SP 800-90A DRBG Service: [1], [3]
DRBG internal state value (C)	SP 800-90A Hash_DRBG Internal state value. 440 – 888 bits	Generated internally.	Never exists the module	Plaintext in volatile memory	Power cycle; API call; or upon uninstantiation of the DRBG.	Internal state value used by the SP 800-90A DRBG Service: N/A
DRBG internal state value (V)	SP 800-90A Hash_DRBG Value of *seedlen* in bits. 440 – 888 bits	Generated internally.	Never exists the module	Plaintext in volatile memory	Power cycle; API call; or upon uninstantiation of the DRBG.	Internal state value used by the SP 800-90A DRBG. Updated during each call to the DRBG. Service: N/A

DRAFT

CGI

Industry Network Protocols

- Probably the worst offenders from a consistency standpoint…
- SSH, SNMPv3 and SRTP are easier
- While TLS and IKE (IPsec) tend to be a little more complex

CGI

Secure Shell (SSH)

Key/CSP	Key/CSP Type	Generation/ Input	Output	Storage	Zeroization	Use
SSH Public and Private Key Pair	RSA: 2048, 3072 4096-bits, ECDSA, DSA: 2048, 3072, 4096-bits	Generated internally / Imported	Public Key: Output during a TLS negotiation Private Key: Does not exit the module.	Persistent. Stored in the modules' non-volatile memory / Ephemeral. Stored in plaintext in non-volatile memory.	Zeroization command / Power cycle.	Key used to authenticate oneself to peer
SSH Session Key	Triple-DES / AES Key size:Triple-DES 168-bits AES 128, 192, 256-bits	Generated internally / Imported	Does not exit the module.	Persistent. Stored in the modules' non-volatile memory / Ephemeral. Stored in plaintext in non-volatile memory.	Power cycle.	to encrypt and authenticate SSH packets
SSH Diffie-Hellman Key Pairs	Diffie-Hellman 2048 bit – 4096 bit/ EC-Diffie-Hellman 2048 bit – 4096 bit. Group 14, 15	Internally generated	Public Key: Output during key agreement Private exponent: Does not exit the module.	Persistent. Stored in non-volatile memory / Ephemeral. Stored in plaintext in non-volatile memory.	Zeroization command / Power cycle.	Generated for SSH key establishment

DRAFT

CGI

Simple Network Management Protocol (SNMPv3)

Key/CSP	Key/CSP Type	Generation/ Input	Output	Storage	Zeroization	Use
SNMPEngine ID	Shared Secret. 32-bits	Specified by operator / Created automatically		Stored encrypted / Stored plaintext NVRAM	Zeroized when overwritten with new engine ID	Unique string to identify the SNMP engine
SNMPv3 Password	Shared Secret / Pre-shared Key. 256-bits				Zeroized when overwritten with new password	Used to derive the SNMP3 Authentication key
SNMPv3 Session Key	AES, Triple-DES CFB-128 bits	Derived Internally using SP 800-135rev1 KDF	Never Exists the Module / Exported in encrypted backup	Ephemeral. Stored encrypted in DRAM/ Persistent. Stored in System FLASH	Power cycle / Zeroize command	Encrypting SNMPv3 packets
SNMPv3 Authentication Key	HMAC-SHA-1	Derived Internally using the SNMP Password	Never Exists the Module / Exported in encrypted backup			Authenticating SNMPv3 packets

DRAFT

CGI

Transport Layer Security (TLS)

Key/CSP	Key/CSP Type	Generation/Input	Output	Storage	Zeroization	Use
TLS Pre-Master Secret Key	Shared Secret, 384-bits	Can be entered encrypted along with RSA public key /Generated Internally.	Can be output encrypted along with RSA public key / Does not exit the module.	Ephemeral. Stored in plaintext in SDRAM.	Power cycle.	Shared secret Exchanged using RSA Key Transport and used to derive the Master Secret
TLS Master Secret	Shared Secret, 384-bits	Computed as part of TLS negotiation according to TLS standard using the Pre-Master Secret and Nonces	Does not exit the module.	Ephemeral. Stored in plaintext in SDRAM.	Power cycle.	Master Secret used to derive the encryption and MAC keys for both ends of a TLS session
TLS Session Authentication Key	HMAC-SHA-1 key, 160 bits	Derived from Master Secret as part of TLS negotiation	Does not exit the module.	Ephemeral. Stored in plaintext in SDRAM.	Power cycle.	MAC key for integrity in one direction
TLS Session Key	AES: 128, 192, or 256 bits; Triple-DES: 168 bits	Derived from Master Secret as part of TLS negotiation	Does not exit the module.	Ephemeral. Stored in plaintext in non-volatile memory.	Power cycle.	Symmetric key used for TLS encryption/decryption of TLS session traffic
TLS Key Exchange Key Pair (Public and Private keys)	RSA: 2048, 3072 bits, ECDSA, DSA: 2048, 3072 bits,	Internally generated / Electronically entered in encrypted form / Electronically entered in plaintext form	Public Key: Output during a TLS negotiation Private Key: Does not exit the module.	Persistent. Stored in the modules' non-volatile memory / Ephemeral. Stored in plaintext in non-volatile memory.	Zeroization command / Power cycle.	Asymmetric key used for negotiation of TLS sessions.
TLS Key Agreement Key	Diffie-Hellman 2048 bit – 4096 bit/ EC-Diffie-Hellman 2048 bit – 4096 bit.	Internally generated	Public Key: Output during key agreement Private exponent: Does not exit the module.	Persistent. Stored in non-volatile memory / Ephemeral. Stored in plaintext in non-volatile memory.	Zeroization command / Power cycle.	Keys used in Key Agreement protocol for Diffie-Hellman key agreement.

DRAFT

IPSec and Internet Key Exchange (IKE) v2

What's Next?

- Ask for some additional review and feedback from CMVP
- SHARE! Post completed template and CSPs on CMUF website
- Reach out to Matt Keller (mkeller@corsec.com) for access to the CMUF site
- Hopefully other vendors and labs will help!!

CGI

A word from the CMVP

CGI

Experience the commitment®

A word from the CMVP

Jennifer Cawthra
CMVP Program Manager
NIST
Email: jennifer.cawthra@nist.gov

Carolyn French
CMVP Lead
CSE
Email: carolyn.french@cse-cst.gc.ca

CGI

Questions, Comments .. Want to contribute?

Ryan Thomas
FIPS 140-2 Program Manager
CGI Global IT Security Labs
Email: ry.thomas@cgi.com

CGI

Thanks!

CGI

Experience the commitment®