Rebranding OpenSSL

A LABORATORY PERSPECTIVE

Jonathan Park / October 13, 2026

Lightship Security

Applus+

THE OPPORTUNITY

Why Rebrand OpenSSL?

The FIPS validation heavy lifting is already done. A rebrand lets a vendor offer OpenSSL's validated FIPS Provider under their own name with a valid Engineering or Enterprise Support contract with the OpenSSL Corporation.

Minimal lab effort

Realistically, the work required from the lab to submit is small — the hard cryptographic work was already validated.

A few details to supply

Vendor name, address, contact information, and the name of the rebranded module — that's the core of what's needed.

An identical module underneath

The resulting module is identical to the source — aside from the vendor name and module name on the certificate.

Contracted with OpenSSL

To rebrand OpenSSL you must have a valid Engineering or Enterprise Support contract in place

WHAT YOU GET

A Vendor Module, based on the OpenSSL FIPS Provider

Naming Convention

<Vendor Module> based on the OpenSSL FIPS Provider

The result:

- A new CMVP certificate which is identical in every functional respect
- Only the vendor name and module name change on the certificate
- The underlying validated cryptography is untouched

BUT — can you sell your products without your OS and hardware listed on the certificate?

Adding Your Operating Environments

Two paths exist to add your specific OS and hardware (OEs) to the new certificate:

VAOE	Vendor Affirmed Operating Environment
OEUP	Operating Environment Update

We'll look closer at the OEUP.

SUBMISSION MECHANICS

Submission Basics & the OEUP Prerequisite

Every submission to the CMVP needs:

01

Security Policy Template

The formatted document that becomes the module's public-facing security policy.

02

A JSON Data File

Structured data merged into the template to produce the final, published policy.

For an OEUP Revalidation

The JSON is a ‘transaction’ JSON — only the sections and fields being updated, not the full submission. What's harder: the tools to build it keep changing.

To add your own OE, you first need your own CAVP certificate:

Test with OpenSSL 3.1.2

The vendor's software/hardware is tested against the same OpenSSL 3.1.2 codebase.

Earn a CAVP Certificate

The CAVP cert (A3548) is owned by The OpenSSL Project.
Vendors need their own as they cannot change A3548

Proceed with the OEUP

With that certificate in hand, the Operating Environment Update can move forward.

As a lab, it's our job to find a way to get a submission package accepted — even when adding a 2nd CAVP certificate “both works and is broken.”

THE TOOL

Automation of the CMVP and Webcryptik

NIST is automating some of the lower effort validations

https://www.nccoe.nist.gov/automation-nist-cryptographic-module-validation-program

Both Rebrands and Operational Environment Updates are automated as part of this ongoing effort.

Labs must use Webcryptik to create the JSON that defines the changes, and to package it with any other required documents — such as the security policy template.

What the lab must do

1. Load your rebranded certificate into Webcryptik
2. Add your algorithm (CAVP) certificate
3. Add the new Operating Environments
4. Update Module Information Structure fields to reference the new certificate

BEFORE YOU START

Questions to Ask the Vendor First

The answers determine exactly what the lab needs to do — so scope this before touching Cryptik.

? Are the original OEs of any value to you specifically?

? Will you keep the original OEs on your certificate?

? Or do you only want your own, specific OEs listed?

8	Windows 10 Pro	Dell Inspiron 7591 2 in 1	Intel i7-10510U	Yes	N/A	3.1.2	
9	macOS 11.5.2	Apple M1 Mac Mini	M1	No	N/A	3.1.2	

A LOOK AT THE WEBCRYPTIK INTERFACE

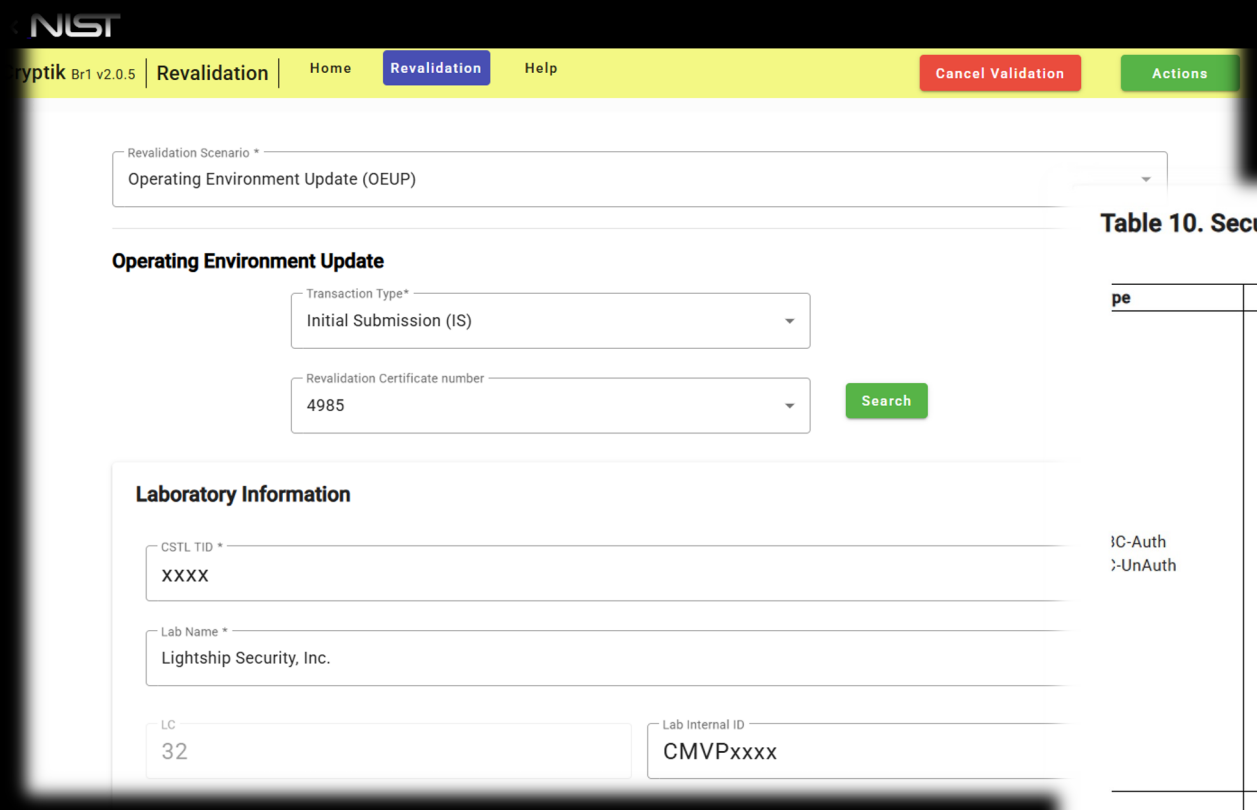


Table 10. Security Function Implementations

pe	Description	SF Properties	Algorithms	Algorithm Properties	Actions
C-Auth -UnAuth	Symmetric Encryption and Decryption	Key Length : 128, 192 and 256 bits Key Length (XTS) : 128 and 256 bits	AES-CBC (A3548)		
			AES-CBC-CS1 (A3548)		
			AES-CBC-CS2 (A3548)		
			AES-CBC-CS3 (A3548)		
			AES-CCM (A3548)		
			AES-CFB1 (A3548)		
			AES-CFB128 (A3548)		
			AES-CFB8 (A3548)		
			AES-CMAC (A3548)		
			AES-CTR (A3548)		
			AES-ECB (A3548)		
			AES-GCM (A3548)		
			AES-GMAC (A3548)		
			AES-OFB (A3548)		
			AES-XTS Testing Revision 2.0 (A3548)		
		SHA-1 : (s = 160) Large Message Sizes: 1, 2, 4, 8gigabytes SHA2 : SHA2-224 (s = 224), SHA2-256 (s = 256), SHA2-	SHA-1 (A3548)		
			SHA2-224 (A3548)		
			SHA2-256 (A3548)		

WEBCRYPTIK: A REALITY CHECK

Webcryptik Tabs That Work As Expected

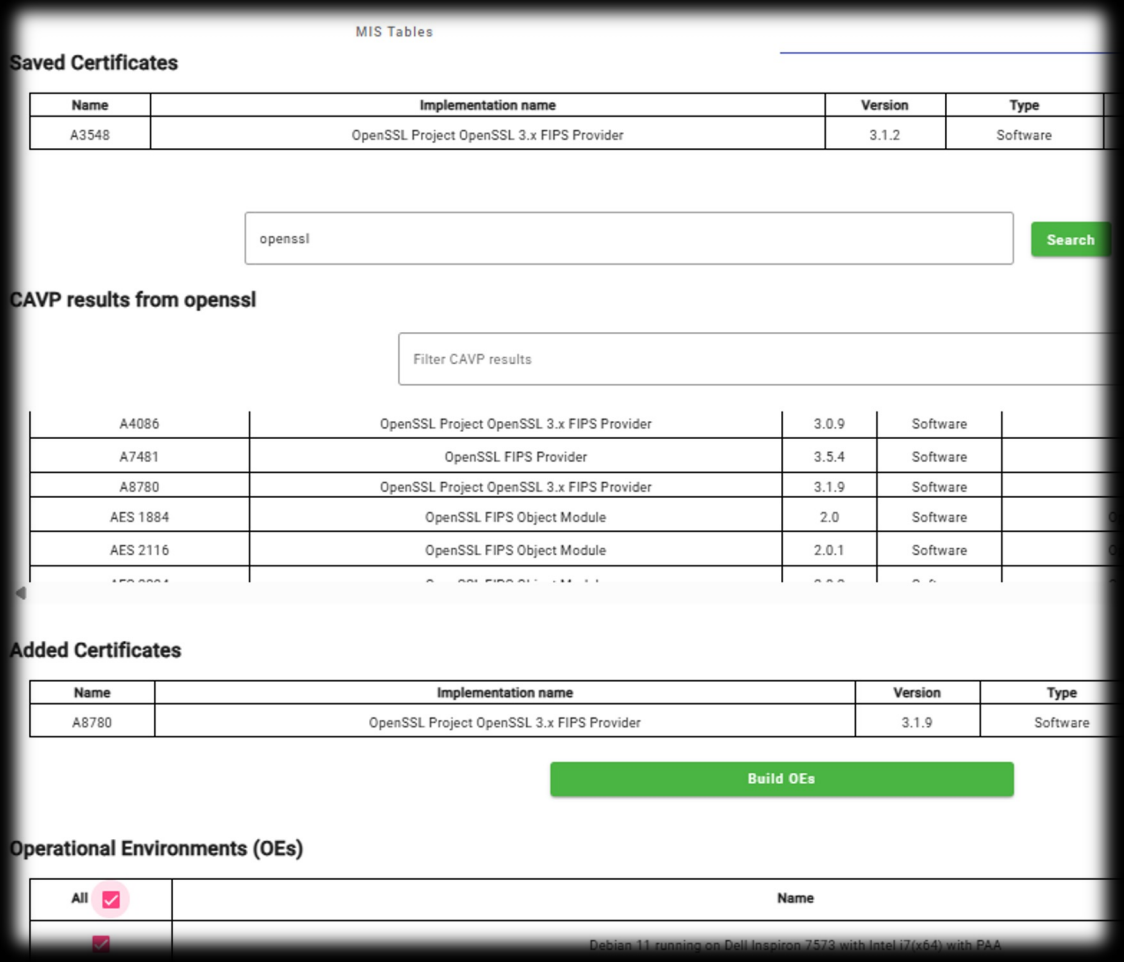
WORKS CLEANLY

CAVP Certs
Adding the new CAVP certificate
Selecting the tested Oes
Selecting and saving algorithms
Removing a CAVP certificate*
*remove its associated OEs and algorithms

MIS Tables — Module Information
Changing the module count*
*depends on Table 2

MIS Tables — Table 2. Tested Module Information
Adding or removing the tested modules

MIS Tables — Table 3. Tested Operational Environments
Adding new OEs
Removing currently listed OEs



WEBCRYPTIK: A REALITY CHECK

Editable — With Real Usability Risk

WORKS, BUT CUMBERSOME

Table 10 — Security Function Implementations (SFI)

New SFIs can be added – but should you?
Existing SFIs can be edited to swap or add an algorithm.
When editing, the information is extremely hard to see.

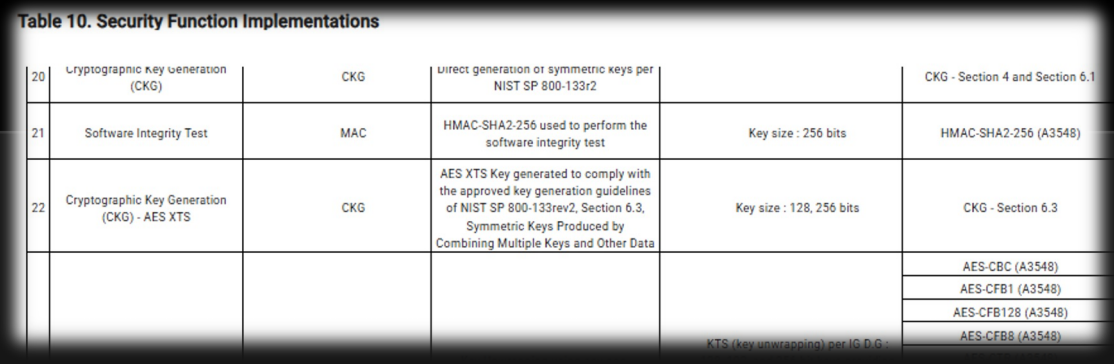


Table 10. Security Function Implementations

20	Cryptographic Key Generation (CKG)	CKG	Direct generation of symmetric keys per NIST SP 800-133r2		CKG - Section 4 and Section 6.1
21	Software Integrity Test	MAC	HMAC-SHA2-256 used to perform the software integrity test	Key size : 256 bits	HMAC-SHA2-256 (A3548)
22	Cryptographic Key Generation (CKG) - AES XTS	CKG	AES XTS Key generated to comply with the approved key generation guidelines of NIST SP 800-133rev2, Section 6.3, Symmetric Keys Produced by Combining Multiple Keys and Other Data	Key size : 128, 256 bits	CKG - Section 6.3
					AES-CBC (A3548)
					AES-CFB1 (A3548)
					AES-CFB128 (A3548)
				KTS (key unwrapping) per IG D.G :	AES-CFB8 (A3548)

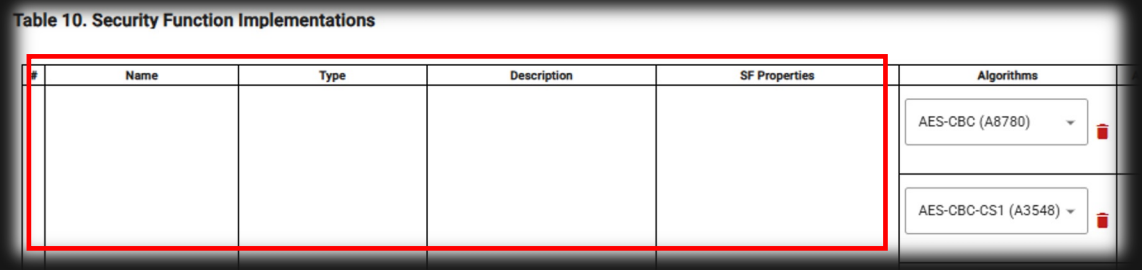


Table 10. Security Function Implementations

#	Name	Type	Description	SF Properties	Algorithms
					AES-CBC (A8780)
					AES-CBC-CS1 (A3548)

Table 26 — Conditional Self-Test

Can be edited, but what one am I editing?
New items can be added, but sorting is an issue.

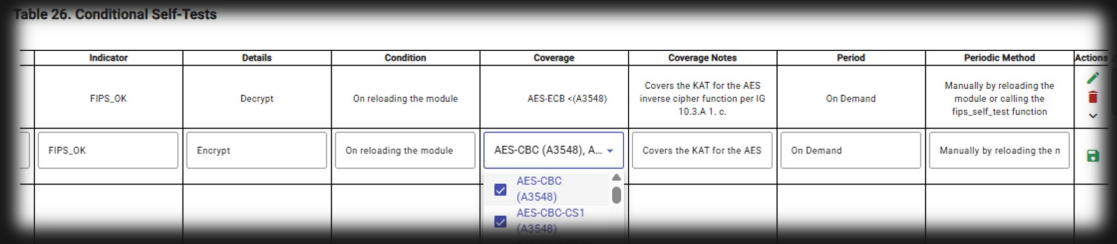


Table 26. Conditional Self-Tests

Indicator	Details	Condition	Coverage	Coverage Notes	Period	Periodic Method	Actions
FIPS_OK	Decrypt	On reloading the module	AES-ECB <(A3548)	Covers the KAT for the AES inverse cipher function per IG 10.3.A 1. c.	On Demand	Manually by reloading the module or calling the fips_self_test function	
FIPS_OK	Encrypt	On reloading the module	AES-CBC (A3548), A...	Covers the KAT for the AES	On Demand	Manually by reloading the n	
			AES-CBC (A3548)				
			AES-CBC-CS1 (A3548)				

WEBCRYPTIK: A REALITY CHECK

Where a Second Certificate Breaks Things

DOES NOT WORK CORRECTLY

Table 6 — Vendor Affirmed Algorithms

The current implementation can be switched to the new certificate
If both certificates are required at once, a second cannot be added.

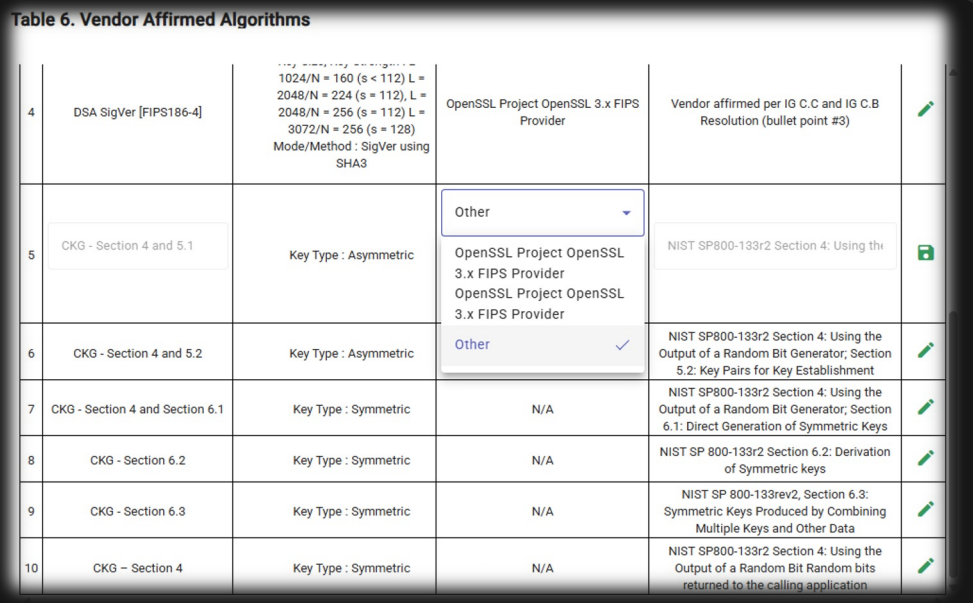


Table 6. Vendor Affirmed Algorithms

4	DSA SigVer [FIPS186-4]	1024/N = 160 (s < 112) L = 2048/N = 224 (s = 112), L = 2048/N = 256 (s = 112) L = 3072/N = 256 (s = 128) Mode/Method : SigVer using SHA3	OpenSSL Project OpenSSL 3.x FIPS Provider	Vendor affirmed per IG C.C and IG C.B Resolution (bullet point #3)	
5	CKG - Section 4 and 5.1	Key Type : Asymmetric	Other	NIST SP800-133r2 Section 4: Using the	
6	CKG - Section 4 and 5.2	Key Type : Asymmetric		NIST SP800-133r2 Section 4: Using the Output of a Random Bit Generator; Section 5.2: Key Pairs for Key Establishment	
7	CKG - Section 4 and Section 6.1	Key Type : Symmetric	N/A	NIST SP800-133r2 Section 4: Using the Output of a Random Bit Generator; Section 6.1: Direct Generation of Symmetric Keys	
8	CKG - Section 6.2	Key Type : Symmetric	N/A	NIST SP 800-133r2 Section 6.2: Derivation of Symmetric keys	
9	CKG - Section 6.3	Key Type : Symmetric	N/A	NIST SP 800-133rev2, Section 6.3: Symmetric Keys Produced by Combining Multiple Keys and Other Data	
10	CKG – Section 4	Key Type : Symmetric	N/A	NIST SP800-133r2 Section 4: Using the Output of a Random Bit Random bits returned to the calling application	

OpenSSL Project OpenSSL 3.x FIPS Provider
OpenSSL Project OpenSSL 3.x FIPS Provider
Other

Table 7 — Non-Approved, Allowed Algorithms

Allows editing the current implementation.
Cannot add another which references
the new certificate

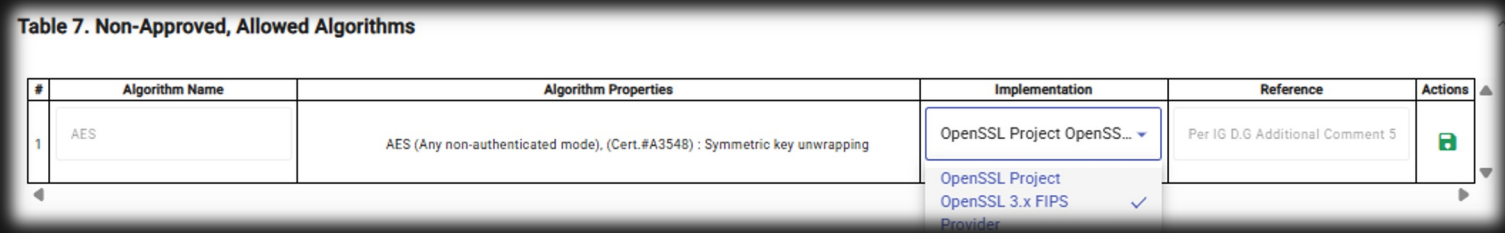


Table 7. Non-Approved, Allowed Algorithms

#	Algorithm Name	Algorithm Properties	Implementation	Reference	Actions
1	AES	AES (Any non-authenticated mode), (Cert.#A3548) : Symmetric key unwrapping	OpenSSL Project OpenSS...	Per IG D.G Additional Comment 5	

OpenSSL Project OpenSSL 3.x FIPS Provider

WEBCRYPTIK: A REALITY CHECK

What Doesn't Work At All

BLOCKED

Cannot update the current certificate reference for the Pre-Operational Self-Tests

To expose this table in the transaction JSON it has to be deleted. There is no ability to edit via cryptik

Table 25. Pre-Operational Self-Tests

#	Algorithm or Test	Test Properties	Test Method	Type	Indicator	Details	Period	Periodic Method	Actions
1	HMAC-SHA2-256 (A3548)	Key Length: 256 bits	KAT	SW/FW Integrity	Success: All self-tests passed (as expected)	MAC (HMAC-SHA2-256, A3548)	On Demand	Manually by reloading the module or calling the fips_self_test function	

THE CONSEQUENCE

Every OEUP Section Needs an Edit

- CAVP Certificate
- Module Info (optional)
- Tested Module Information
- Tested Operational Environments
- Vendor Affirmed Algorithms
- Non-Approved, Allowed Algorithms
- Security Function Implementations
- Pre-Operational Self-Tests
- Conditional Self-Tests

THE LAB'S WORKAROUND

As of Today, the Only Path Forward

1. Make edits across all sections in Webcryptik, so each affected section appears in the transaction JSON
2. Manually edit that JSON — adding sections, possibly referencing a full submission module for anything that was completely removed
3. Confirm the correct fields are actually present before updating any information
4. Re-import the corrected JSON back into Webcryptik so the changes display correctly
5. Build and submit the final submission package

EXPERIENCE OVER TIME

The First Time Is the Hardest — Then What?

The First Attempt

- Time spent studying the JSON structure
- Adding sections by hand, one at a time
- Checking a full submission for sections removed entirely
- Iterating until the JSON loads cleanly into Cryptik

Over Time

- The sensible move is to script the repeated changes
- Webcryptik itself keeps changing across versions
- Scripts needs ongoing testing and maintenance
- There's no ‘solve it once’ state — it's a living process

FROM THE FIELD

I've worked on submissions where no edits were possible in Webcryptik at all — yet Lightship was still expected to produce the correct transaction JSON and build the package.

Working directly with the CMVP, and demonstrating what wasn't exposed in the tooling, is what got us to the state we're in today.

CLOSING THOUGHT

A Good Starting Point — With More Work Ahead

For most vendors, a straight rebrand of OpenSSL is a good starting point.

But the moment you continue on to an OEUP, there is more work involved — on the lab's side — than most vendors realize.

- Rebrand alone → minimal lab effort
- Add your own OEs (OEUP) → significant, ongoing tooling workarounds

Questions?

Rebranding OpenSSL — A Laboratory Perspective

Associate FIPS Tester: Jonathan Park – jon.park@lightshipsec.com
Laboratory Manager: James Ramage – james.ramage@lightshipsec.com